ESTUDIO DE LOS MOTORES DE SÍNTESIS DEL HABLA

AUTOR DEL PROYECTO:

VICENTE PÉREZ LÓPEZ

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS, ESPECIALIDAD MULTIMEDIA

DIRECTOR DEL PROYECTO:

D. MANUEL AGUSTÍ I MELCHOR

DEPARTAMENTO DE INFORMÁTICA DE SISTEMAS Y COMPUTADORES

6 DE JUNIO DE 2011

CONTENIDOS

1_ INTRODUCCIÓN

El objetivo del proyecto es el estudio y aplicación de los motores de síntesis del habla, así como la adaptación de textos de diferente procedencia para ser automáticamente adecuados y leídos.

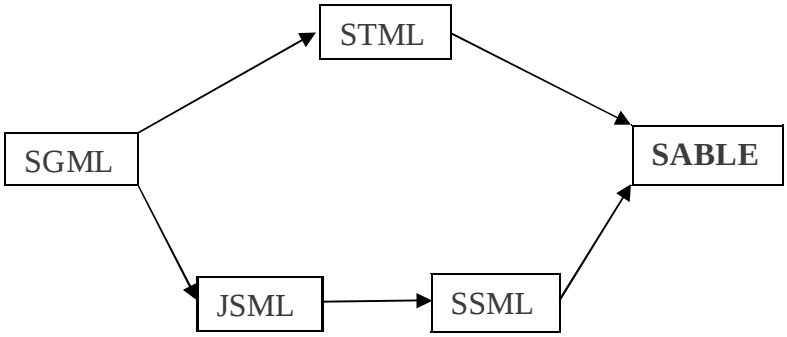
Las aplicaciones de la síntesis del habla pueden ser diversas, desde la lectura de obras literarias, hasta reforzar información visual, como puedan ser los números de los turnos de un establecimiento o entidad, o los horarios y destinos en estaciones de medios de transporte. Pero esto es de vital importancia en el caso de las personas invidentes, cuando la información auditiva deja de ser un complemento para ser muchas veces la única de que se dispone. Un aspecto crucial para ellas es el del acceso a la información, pues mayormente sólo tienen acceso a los medios auditivos, principalmente la radio. Así pues una importante aplicación es la de abrir el acceso a la prensa de los medios digitales escritos y es en este sentido en el que se centrará principalmente el estudio del proyecto. Siempre utilizando software libre, defendiendo el derecho a la libre información.

El resultado principal del estudio será la creación de una aplicación que tras ejecutarse obtenga las noticias del día de la o las webs previamente indicadas y automáticamente las convierta a ficheros convenientemente preparados para su interpretación con una voz sintetizada. Para esto vamos a determinar cómo resolveremos las siguientes cuatro cuestiones: Cómo se obtendrán las noticias; cómo se prepararán estas para ser leídas; cómo serán interpretadas mediante voz; y cómo se integrará todo esto en una aplicación.

Para la obtención de noticias utilizaremos la sindicación mediante RSS. El RSS[4] (Really Simple Sindication) proporciona un mecanismo para obtener archivos de texto formateados. Estos archivos, aunque son usados mayoritariamente para mostrar su contenido en otras webs y aplicaciones, podemos utilizarlos para analizarlos y derivar de ellos archivos preparados para ser leídos. El hecho de que los ficheros de los RSS tengan formato de etiquetas XML[3] nos facilitará la tarea de desglose de éstos, así como la automatización de su conversión a otro formato adecuado para la lectura de motores de síntesis; Este formato será SABLE.

SABLE[1] es un lenguaje de marcado basado en XML, desarrollado por la universidad de Edimburgo en colaboración con Sun, ATT y Bell Labs, que nace con la intención de convertirse en un estándar de ficheros de representación de la síntesis TTS (Text To Speech, texto a habla). Fue creado a partir de tres lenguajes de marcado anteriores: SSML (Speech Síntesis Markup Language), STML (Skunk

Template Markup Language) y JSML (Java Speech Markup Language), desarrolladas por el consorcio W3C.



Este lenguaje permite controlar aspectos de la lectura como la voz utilizada, la velocidad, el volumen, la entonación y las pausas, con lo cual proporciona unos mecanismos básicos para humanizar en lo posible la lectura sintetizada. Como está basado en etiquetas XML, la conversión a SABLE podrá realizarse bien por código desde funciones de la propia aplicación, o bien mediante un fichero XSLT (eXtensible Stylesheet Language Transformations). XSLT[3,5] es un estándar del consorcio W3C que proporciona hojas de estilo que permiten transformar documentos basados en etiquetas de marcado en otros. En nuestro caso nos permite convertir los XML provenientes de las sindicaciones RSS en ficheros SABLE, listos para ser interpretados por el motor de síntesis: Festival.

Festival[2] es un motor de síntesis del habla, de libre distribución, desarrollado inicialmente por la Universidad de Edimburgo pero con importantes aportes posteriores de otros centros de enseñanza. Estuvo concebido originalmente para interpretar textos en inglés, pero posteriormente la comunidad ha extendido su funcionalidad a otros lenguajes incluído el castellano, y ha aportado nuevas voces gracias al proyecto Festvox (un conjunto de herramientas para facilitar la construcción de voces sintéticas). Aunque existen numerosos motores de síntesis, hemos elegido Festival por varios aspectos, como la portabilidad, pues en teoría es compatible con cualquier sistema Unix que utilice el entorno de escritorio Gnome (principalmente Debian, Ubuntu, Red Hat, y Fedora). Además es posible su integración en aplicaciones ya que ofrece librerías para Java y C. Y por último, es interesante por su versatilidad, pues puede interpretar tanto directamente texto plano (utilizando los parámetros por defecto), como ficheros en diversos lenguajes de marcado, como es el caso de nuestro elegido, el SABLE.

Para todo esto crearemos una aplicación en lenguaje C. Este será el lenguaje elegido por su potencia, familiaridad y compatibilidad con las librerías disponibles para el motor de síntesis comentado. La intención es crear una interfaz en la que sea prescindible la información visual, siendo un mero refuerzo, proporcionando una funcionalidad completa mediante la presentación de las opciones disponibles mediante voz sintética y esperando alguna acción por parte del usuario.

2_ ESTADO DEL ARTE

2.1_ MARCO

En la actualidad, tanto la síntesis Text-To-Speech (texto-a-habla), como el reconocimiento Speech-To-Text (habla-a-texto), se hallan considerablemente extendidos en diversos ámbitos de la vida cotidiana en que entre en juego la interacción hombre-máquina. ¿Quién no ha tenido que decir "sí" para elegir una opción, o ha elegido escuchar su saldo telefónico, con una teleoperadora virtual? Lo cierto es que las ventajas de la conversión de doble sentido entre el texto y la voz son innegables. Aunque quizás el ejemplo recién presentado puede despertar en más de uno cierto espíritu ludita[6], haciendo pensar que únicamente es un recurso más para ahorrar mano de obra en las compañías, la realidad es que es un avance muy útil y en ocasiones imprescindible en varias tecnologías implantadas hoy en día. Algunos ejemplos actuales nos harán ver esta importancia en los dos ámbitos de aplicación principales:

-Entornos de trabajo limitados o de distracción impermisible:

-STT (Speech-To-Text): Por ejemplo, es delicado el uso de la telefonía móvil mientras se conduce un vehículo, donde la posibilidad de marcar por voz un número de teléfono nos evita el riesgo (y delito, dicho sea de paso) de tener que apartar la vista de la carretera y la mano del volante.

-TTS (Text-To-Speech): Siguiendo con los vehículos, también es importante el texto-a-habla por ejemplo en el tema de los GPS, donde cada vez más éstos nos indican por voz incluso los nombres de las calles que debemos tomar, evitando de nuevo peligrosas distracciones por mirar la pantalla del dispositivo (GPS Navigon, Copilot, Google Maps Navigation, etc).

Un inciso importante: Quizás este es un buen punto para dejar claro el hecho de que no todo lo que escuchamos de un aparato que "nos habla" es TTS; No es TTS un contestador que nos reproduce ficheros de audio con los dígitos pregrabados del número de teléfono al que llamamos, ni es TTS los GPS que se limitan a reproducir las diez o quince instrucciones pregrabadas típicas de las acciones que debemos realizar en cada situación... Podríamos decir que la gran diferencia es que los mensajes pregrabados son una solución sencilla cuando un número concreto y reducido de respuestas es suficiente, y el TTS en el caso contrario, cuando los mensajes son impredecibles y se deben interpretar dinámicamente conforme van surgiendo.

-Personas con discapacidades visuales y/o motoras:

-STT: Por ejemplo las personas con dificultades de movimiento en las manos, tienen a su disposición sistemas de dictado que les permiten plasmar las palabras que van dictando en documentos escritos verificables en tiempo real. Fue especialmente popular el sistema VoiceType de IBM[7] en los años 90, y se ha extendido hasta nuestros no sólo como aplicaciones para los Pcs tradicionales (p.e. Express Dictate) sino también a otros dispositivos como por ejemplo grabadoras portátiles que almacenan en formato de texto lo que se les dicta por voz (gamas Phillips[8], Olympus[9],...).

-TTS: Las personas invidentes no tienen acceso a la información escrita que ofrece la pantalla del ordenador, por lo que son necesarios programas que interpreten esos textos para que esa información esté a su alcance. A este último ejemplo de la división que hemos propuesto es al que va especialmente destinada la aplicación principal de este proyecto, el lector de noticias "web-a-voz".

También es interesante destacar que el uso de TTS no sólo permite que llegue información a la persona discapacitada, sino que en ocasiones lo que permite es que sea la propia persona la que se pueda comunicar con los demás. Es especialmente famoso el caso del astrofísico Stephen Hawking[10], al cual una traqueotomía le impide hablar y una esclerosis lateral amiotrófica le impide moverse más allá de ligeros gestos con la cabeza. Es con estos gestos con los que, mediante un complejo sistema[11], selecciona sucesivamente de entre 3.000 palabras para ir componiendo lentamente oraciones que un sintetizador incorporado va leyendo en voz alta. Como curiosidad destacar que la voz de dicho sintetizador utiliza una voz con acento estadounidense, detalle que el doctor, entre bromas, reconoció que inicialmente le disgustó (Stephen Hawking es inglés).

En el siguiente apartado haremos un repaso por algunos de los mecanismos actuales que se esconden tras los sistemas de síntesis de texto-a-habla.

2.2_ TECNOLOGÍA SUBYACENTE

Tal y como avanzábamos en la introducción, para que la síntesis del habla sea posible se necesitan dos elementos: el primero, un formato de representación que sirva como entrada, que puede ser desde algo simple como el texto plano, hasta un lenguaje de marcado con parámetros que modifiquen aspectos como la prosodia[12] de la voz. El segundo, un motor de síntesis de texto-a-habla, encargado de interpretar el texto de entrada mediante una voz sintética. Nótese que aunque es habitual encontrar el término "texto-a-voz", ésta no la consideramos una traducción precisa,

siendo el original anglosajón “text-to-speech” (texto-a-habla o texto-a-discurso) y no “text-to-voice”, por lo cual intentaremos utilizar el término que consideramos más preciso.

2.2.1_ LENGUAJES DE MARCADO

- VoiceXML[13] (Voice Extensible Markup Language): Es un estándar del consorcio W3C, basado en etiquetas XML, creado para proporcionar a los desarrolladores de aplicaciones y páginas web un conjunto de etiquetas que sirvan tanto para indicar que se debe sintetizar cierto texto a voz, como para esperar órdenes de voz por parte del usuario que se deban interpretar. Los archivos creados con este formato se procesan mediante un navegador de voz o Voice Browser[14], de la misma forma que un navegador convencional procesa los archivos HTML[15]. En el caso del texto-a-habla los navegadores de voz pueden, bien reproducir ficheros de audio pregrabados, bien utilizar un motor de síntesis del lenguaje de entre los disponibles. Algunos ejemplos de estos navegadores de voz son el OpenVXI[16], el JvoiceXML[17], y el PublicVoiceXML[18].

Los navegadores de voz se usan en diferentes aplicaciones como el acceso por voz a bases de datos, recursos de telefonía (p.e. marcación por voz), o acceso por voz al correo electrónico.

- JSML[19] (Java Speech Markup Language): Lenguaje de marcado ideado para suministrar a los sintetizadores de voz ciertos detalles que ayudan a humanizar en lo posible la lectura del texto al que acompañan. Se compone de diversas etiquetas XML que podemos dividir en tres grupos:
Estructurales: Para señalar y separar párrafos y oraciones → <jsml> y .
De producción: Definen propiedades auditivas de la síntesis → <voice>, <sayas>, <phoneme>, <emphasis>, <break> y <prosody>.
Misceláneos: Controles embebidos en el texto para habilitar controles específicos del sintetizador → <marker> y <engine>.

- SSML[20] (Speech Síntesis Markup Language): Lenguaje de marcado sucesor directo de JSML. Hereda la estuctura de éste y lo complementa con algunas nuevas etiquetas:
<audio>: permite la inserción de ficheros de audio pregrabados.
<paragraph> y <sentence>: nuevas etiquetas estructurales.
<sub>: sustituye un texto por otro para mejorar la pronunciación.
<say-as>: indica que se requiere un modo de lectura diferente del habitual, como por ejemplo ante un formato de fecha.

- **STML[21] (Skunk Template Markup Language)**: Otro lenguaje de marcado de texto para marcar el texto que se suministra a los sintetizadores de voz. El conjunto de etiquetas está prácticamente en su totalidad contenido en su sucesor SABLE, con algunas

excepciones como <phonetic> u <omitted>.

En la sección 4.1 hablaremos del formato SABLE, sucesor de los dos últimos estándares citados (SSML y STML) tal como mostrábamos en la introducción, que es el elegido para la versión inglesa de nuestro programa principal (explicaremos en su momento por qué no para la española).

2.2.2_ MOTORES DE SÍNTESIS DEL HABLA

En la actualidad existen tres grandes grupos de motores de síntesis[22] del habla:

- Basados en Formantes[23]: Sintetizadores que producen la onda sonora a partir de un modelo acústico variando parámetros físicos como el pitch[24], sin intervención en ningún momento de segmentos reales de voz humana. Este tipo de sintetizadores tienen sentido en condiciones muy restrictivas de hardware (p.e. sistemas embebidos[25]), pero en general no son usados puesto que producen una voz completamente robotizada.

- Concatenativos: Estos sintetizadores se diferencian de los anteriores en que generan la voz artificial enlazando segmentos pregrabados de un locutor humano. Dentro de esta técnica existen diversas variantes, entre las que se encuentran los sintetizadores basados *en "difonemas*" cuya unidad mínima es la concatenación de cada par de fonemas, que son grabados y almacenados en una base de datos, y "*sintetizadores basados en selección de unidades*", que también registran una base de datos de segmentos pregrabados, pero en este caso no se limitan a difonemas, pudiendo encontrar desde fonemas simples hasta oraciones completas. En este grupo se encuentra nuestro motor escogido, Festival.

- Articulatorios: Este tipo de sintetizadores convierten cadenas de texto en descripciones fonéticas, ayudados por un diccionario de pronunciación, reglas de caracteres-sonidos, ritmo y modelos de entonación. Posteriormente transforman las descripciones fonéticas en parámetros de articulación de bajo nivel para el sintetizador que son usados para generar un modelo de articulación vocal humano produciendo una salida conveniente para los dispositivos de salida de audio.

A continuación hacemos un repaso de algunos de los motores de síntesis del habla que podemos encontrar en la actualidad:

- Verbio[26] **TTS**: Motor de síntesis texto-a-habla comercial, propiedad de la española Verbio Speech Tecnologies, que dispone de voces de ambos géneros en varios idiomas. Forma parte de un conjunto de herramientas tanto de TTS como de STT de las que

también forman parte:
Verbio ASR[27]: Reconocedor del habla (STT).
Verbio VoiceXML: Navegador de voz (véase el formato VoiceXML).
Verbio Speaker Id: Identificador de propietario de voz como mecanismo de seguridad (acceso a terminales, apertura de puertas, etc).
Verbio WordSpotting: Buscador de palabras clave en ficheros de audio.
Verbio VoiceWeb: Navegación web mediante la voz sin necesidad de usar ratón ni teclado.

- FreeTTS[28]: Motor de libre uso y comercialización, escrito totalmente en JAVA por la Carnegie Mellon University y basado en Flite[29], un pequeño motor de síntesis en tiempo de ejecución derivado de Festival, que de forma similar a éste sintetiza por concatenación de difonemas. Ofrece compatibilidad con voces inglesas estadounidenses y británicas en ambos géneros, entre ellas las provenientes de los proyectos FestVox[30] de Festival y MBRola[31]. Además soporta parcialmente JSAPI[32] 1.0. Funciona en sistemas Linux, Mac OS/X, Win32 y Solaris.

- Loquendo[33] TTS: Motor de síntesis comercial creado por la empresa italiana del mismo nombre, propiedad de Telecom Italia Lab, basado en sintetización por selección de unidades. Es un motor de gran calidad y popularidad, especialmente con las lenguas latinas. Forma parte de un conjunto de productos mayor entre los que se encuentran un motor de habla-a-texto y un identificador de propietario de voz.

- GNUSpeech[34]: Sintetizador de voz articulatorio ofrecido con licencia GPL V2, disponible para Linux y para Mac OS/X, que actualmente sólo tiene disponible la lengua inglesa. Lo destacado de este sintetizador es que se basa en un modelo de tubo de resonancia (TRM) imitando el tracto vocal humano para generar los sonidos, que busca alcanzar el máximo realismo teniendo en cuenta el comportamiento de las cavidades nasal y oral, así como la impedancia al aire de la nariz y los labios.

- Nuance Realspeak[35] TTS: Motor de síntesis propiedad de la empresa norteamericana Nuance, basado en la síntesis concatenada por selección de unidades. Forma parte de una gran gama de productos para particulares y empresas. Dispone de un gran abanico de lenguas disponibles.

Además de estos existen muchos otros motores de voz, como MBRola, eSpeak[36], SodelsCot[37], Lernout y Hauspie[38], NeoSpeech VoiceText[39] o Fonix Speech[40], entre otros.

Al igual que hicimos con el formato SABLE, emplazamos la descripción de nuesto motor escogido, el Festival Speech System, a su sección correspondiente (3.1).

2.2.3_ ALGUNOS PROYECTOS DESTACABLES

- DAISY[41] (Digital Accesible Information System): Es el consorcio internacional que promueve la universalización de estándares con un conjunto de características para producir libros hablados digitales, para personas con limitación visual y problemas de acceso a la lectura, que permite que los libros producidos sean consultados, marcados, retomados, y transportados con las mismas ventajas de los libros convencionales. Estos libros se leen con software y hardware específicos que permitan aprovechar las características de navegabilidad a varios niveles que ofrecen: por páginas, frases, marcas, capítulos o subcapítulos.

- Proyecto VOICES[42]: Proyecto para difusión de la comunicación en Senegal y Mali, llevado a cabo por el considerado padre de Internet, Tim Berners-Lee[43], junto con la organización World Wide Web Foundation[44]. El objetivo de este proyecto es ofrecer a las familias y a la juventud de dichos países con pocos conocimientos informáticos una forma de acceder a internet a través de la voz, especialmente a aquellas webs referentes a la agricultura o a la salud, principales preocupaciones en la población de ambos países.

- Proyecto Asterisk[45]: Asterisk es un programa con licencia GPL creado por Mark Spencer, fundador de Digium, que nace con la intención de proporcionar las funcionalidades de una central telefónica hasta ese momento sólo disponibles en sistemas de alto coste, como son: buzón de voz, conferencias, distribución automática de llamadas, y la que nos interesa por el tema que nos ocupa, el IVR[46].

IVR (Interactive Voice Response, respuesta de voz interactiva) es un sistema que es capaz de recibir una llamada e interactuar con el humano en los dos sentidos de la comunicación. Por un lado, es capaz de usar un motor de síntesis TTS para leer en voz alta cualquier información disponible en una base de datos. Por otro lado, cuando sea necesario puede procesar respuestas humanas convirtiéndolas a texto con un reconocedor de voz (ASR), típicamente “sí”, “no”, o algún número para seleccionar una opción concreta. En las empresas grandes esta interacción tiene el fin de enrutar la llamada hacia el departamento más adecuado sin necesitar la intervención humana, ahorrando tiempo y personal.

Uno de los motores de síntesis del habla más utilizados para ser integrados en Asterisk es precisamente Festival, el mismo que hemos elegido nosotros para el presente proyecto.

2.3_ PROPUESTA

El resultado final de este proyecto será la realización de algunas aplicaciones con el objetivo común de facilitar,

principalmente a las personas invidentes o con deficiencias visuales severas, el acceso a información escrita de diversas procedencias mediante la interpretación hablada de ésta con un motor de síntesis del habla, el Festival Speech Synthesis System. Son tres aplicaciones, aunque las dos primeras tienen el mismo objetivo conseguido de dos maneras distintas. Son las siguientes:

- Web-a-Voz: Esta aplicación pretende acceder a los RSS de algunas de las webs de noticias nacionales, para obtener de ellas el texto referente a las noticias, filtrar el contenido que no se debe interpretar y adecuarlas para que el motor de síntesis las lea de la manera más afín posible a como lo haría una persona. Esta versión es en castellano.

- Web-2-Voice: Esta aplicación es la versión inglesa de la arriba mencionada, y aunque el resultado final apreciable es el mismo (con la salvedad del idioma), mostraremos mecanismos alternativos para obtenerlo.

- E-Narrador: La tercera de las aplicaciones propuestas será un lector de documentos. El usuario podrá introducir cuantos documentos desee en una carpeta determinada para que posteriormente en ejecución se ofrezca comenzar a leer el que se desee. La característica particular de este narrador electrónico será la posibilidad de retomar la lectura de cada documento por donde se dejó en la anterior ocasión.

3_ FESTIVAL SPEECH SYNTHESIS SYSTEM

3.1_ ¿QUÉ ES?

Festival Speech Synthesis System es un motor de síntesis de código abierto creado por la Univesidad de Edimburgo, ofrecido bajo un tipo de licencia denominado "X-11"[47] que permite su uso libremente, tanto para uso particular como comercial. Ofrece síntesis concatenativa, con un algoritmo robusto basado en concatenación de difonemas, y varias opciones alternativas como "Unit Selection", "HTS" o ClusterGen. Su núcleo está programado en C++ aunque se puede configurar utilizando Scheme (lenguaje de programación similar a LISP[48]).

Originalmente sólo estaba el inglés como voz disponible, pero posteriormente la comunidad ha desarrollado voces en castellano, finlandés, italiano, checo, hindi, polaco, marathi, telugu y ruso. Esto ha sido posible gracias a las herramientas y a la completa documentación del proyecto Festvox, que busca facilitar a la comunidad la creación de voces de calidad para Festival en cualquier idioma.

Lo hemos seleccionado para ser el motor de síntesis de nuestro proyecto por varias razones:

- Es un motor de código libre, lo que nos da total libertad para desarrollar y distribuir nuestras aplicaciones.
- Le sigue una comunidad activa que ha desarrollado voces de calidad en nuestro idioma.
- Es compatible con cualquier sistema Unix que utilice el entorno de escritorio Gnome (principalmente Debian, Ubuntu, Red Hat, y Fedora).
- Proporciona una completa API para C++ que permite su integración en nuestras aplicaciones.
- Existen ejemplos disponibles que nos permiten escuchar cómo es en funcionamiento y que nos animan a utlizarlo. Por ejemplo, el siguiente enlace nos muestra una página web que nos permite escuchar cómo suena el motor Festival con las voces castellanas desarrolladas por la Junta de Andalucía[49], de las que hablaremos más tarde:

http://vozme.com/index.php?lang=es

3.2_ INSTALACIÓN Y CONFIGURACIÓN

En este apartado vamos a indicar cómo instalar el motor de síntesis Festival. En el momento de la realzación de este proyecto, la versión disponible del programa es la 1:2.0.95-beta-2ubuntu1, por lo cuál no podemos garantizar que las aplicaciones

desarrolladas funcionen de la forma esperada con versiones posteriores. También podría haber diferencias en la configuración y en las rutas de los ficheros si se usa una versión diferente del sistema operativo en que se instalan. En nuestro equipo es la versión 10.10 de Ubuntu.

Procedamos a la instalación. Para ello escribimos en un terminal la siguiente línea:

apt-get install festival festival-dev festvox-ellpc11k festvox-rablpc16k

NOTA: Obviamente, como siempre que instalamos aplicaciones, será necesario acceder préviamente como superusuario, mediante la orden "su" e introduciendo la contraseña correspondiente.

Si la instalación tiene éxito, habremos instalado los siguientes paquetes y sus dependencias:

festival contiene el núcleo del sintetizador y su intérprete de órdenes.
festival-dev contiene el conjunto de librerías y cabeceras para el desarrollo de aplicaciones que utilicen Festival.
festvox-ellpc11k contiene la voz castellana principal para Festival.
Festvox-rablpc16k contiene la voz inglesa británica masculina para Festival.

Además instalaremos las dos voces en castellano de mayor calidad que hay disponibles. Fueron desarrolladas por dos empresas sevillanas, Indisys y MP-Sistemas, en respuesta a una petición de la Junta de Andalucía para dotar a su distribución GuadaLinex de un sintetizador de voz. Son una voz masculina (Pedro) y una femenina (Silvia), y ambas están disponibles bajo licencia libre.
Se pueden obtener desde la página oficial en las siguientes direcciones:

Voz masculina:
http://ftp.cica.es/Guadalinex/guadalinex_edu/pool/main/f/festvox-palpc16k-cga/festvox-palpc16k-cga_0.1-3_all.deb

Voz femenina:
http://ftp.cica.es/Guadalinex/guadalinex_edu/pool/main/f/festvox-sflpc16k/festvox-sflpc16k_1.0-cga0_all.deb

(También están disponibles en el anexo de esta memoria, en la carpeta "VOCES")

Como puede verse, estos paquetes no corresponden con la distribución actual de Ubuntu, sino a Debian, por lo que no los podemos instalar de la forma habitual. Pero no hay problema, pues tenemos tres alternativas igual de válidas para su instalación:

1) Haciendo doble clic sobre los paquetes en el entorno gráfico, lo cual nos abrirá el "Centro de Software de Ubuntu" que se encargará de la instalación fácilmente.

2) Utilizando el instalador dpkg[50].

3) Utlizando el instalador gdebi[51]. Esta es la opción que utilizamos nosotros. Para ello, primero se debe instalar el paquete de gdebi de la siguiente manera:

apt-get install gdebi-core

Y a continuación instalamos cada voz por separado así, suponiendo que el directorio actual en el que nos encontramos en el terminal es el que contiene ambos paquetes .deb:

Para la voz masculina:
gdebi festvox-palpc16k_1.0-1_all.deb

Para la voz femenina:
gdebi festvox-sflpc16k_1.0-1_all.deb

También es un buen momento para, si no se tiene aún, instalar el G++, el popular compilador de C++, en nuestro caso la versión 4.4.5:

apt-get install g++

Tras estas instalaciones, ya tendremos lo necesario para hacer funcionar Festival.

A continuación echaremos un vistazo al fichero de configuración del programa. Para ello (siempre como super-usuario) teclearemos en consola:

gedit /etc/festival.scm

Lo cual nos abrirá dicho fichero en modo edición, listo para editarlo. Este es el lugar donde deberíamos poner cualquier orden que queramos que se ejecute automáticamente en el mismo instante en que iniciamos Festival. Ahora mismo, el fichero debería contener las siguientes líneas:

(set! voice_default 'voice_JuntaDeAndalucia_es_pa_diphone)
(set! voice_default 'voice_JuntaDeAndalucia_es_sf_diphone)

Aunque el orden dependerá de cuál de las voces de la Junta de Andalucía hemos instalado primero. Esto es así, porque los paquetes de estas voces no sólo instalan la voz en sí sino que además las establecen como voces predeterminadas. El orden de ejecución es de arriba a abajo, por lo que la segunda línea anula la primera, con lo cual en el ejemplo mostrado la voz de la segunda línea sería la utilizada por defecto (la femenina). No obstante, no es deseable que ninguna de estas voces sea la voz por defecto, siendo mejor utilizarlas cuando lo indiquemos explícitamente. Para ello, no tenemos más q borrar estas líneas, o comentarlas añadiendo ; delante si se prefiere conservarlas.

El establecimiento de la voz por defecto es sólo una de las utilidades de este fichero de configuración. De hecho, la más importante es otra: cargar un controlador de audio adecuado a nuestro equipo. Aquí mostramos un ejemplo de las líneas que deberíamos añadir al fichero "festival.scm" para indicar que queremos que Festival utlice el controlador de sonido PulseAudio[52]:

(Parameter.set 'Audio_Command "paplay $FILE")
(Parameter.set 'Audio_Method 'Audio_Command)
(Parameter.set 'Audio_Required_Format 'snd)

En cualquier caso tampoco indicaremos nosotros manualmente qué controlador utilizar, sino que dejaremos que sean las aplicaciones que hemos diseñado las que detecten cuál es el controlador adecuado, evitando a los usuarios la molestia de tener que aprender la sintaxis de Festival o comprobar los controladores de su equipo. Explicaremos debidamente la función destinada a ello en la sección 5.1.3. Así pues, recomendamos que el fichero de configuración quede totalmente en blanco (o con todas sus líneas comentadas con ;).

En el siguiente apartado echaremos un vistazo a las ya vistas en el fichero de configuración y otras órdenes de Festival.

3.3_ PRIMEROS PASOS

Vamos a probar por primera vez cómo suena nuestro recién instalado motor de síntesis Festival. Para ello, desde consola, lanzamos la siguiente orden, que ejecutará el intérprete de órdenes de Festival preparado para utilizar la voz principal en castellano (la festvox-ellpc11k que habíamos instalado):

festival --language spanish

Tras unas líneas que nos presentan el programa, podremos ver que tenemos el siguiente prompt que nos indica que estamos ejecutando el intérprete de Festival:

festival>

Ahora es cuando podremos escribir cualquiera de las órdenes de Festival disponibles. Probemos la que nos permite leer cualquier texto que le pasemos como parámetro:

(SayText "hola mundo")

Idealmente, deberíamos haber escuchado la voz principal española diciendo el texto que le hemos indicado, “hola mundo”. Pero si no escuchamos nada no hay de qué preocuparse, lo más probable es que el controlador por defecto que ha intentado utilizar Festival no sea válido, con lo cual para escuchar este ejemplo tendríamos que ejecutar préviamente la línea correspondiente a nuestro controlador, o incluirla en el fichero de configuración que habíamos presentado. Pero esto no será necesario para que funcionen nuestras aplicaciones porque, como ya dijimos antes, ellas mismas se encargarán de detectar el controlador adecuado.

Las órdenes que acepta este intérprete son:

(voice_XXXXX) [siendo XXXXX el nombre de una voz instalada]
Establece que se utilizará la voz indicada para las siguientes lecturas que se efectúen.

(SayText “XXXXX”) [siendo XXXXX una cadena de texto cualquiera]
Realiza la conversión TTS utilizando la última voz cargada o, en su defecto, la predeterminada, tal como hemos visto en el ejemplo.

(load ”XXXXX.scm”) [siendo XXXXX la ruta del fichero .scm]
Carga un fichero .scm (Scheme) con un número cualquiera de líneas de órdenes de Festival y las ejecuta secuencialmente, de forma análoga a como hace el programa nada más ejecutarse con el fichero de configuración “festival.scm”.

(exit)
Abandona el intérprete de ordenes de Festival. Con Control+D se consigue el mismo efecto.

También se puede utilizar Festival sin necesidad de entrar en el intérprete de órdenes, utilizando el shell. Por ejemplo, podemos reproducir la frase “hola mundo” con un pipeline que comunique la salida de la instrucción *echo* con la entrada de Festival:

echo "hola mundo" | festival --tts --language spanish

O bien podemos suministrarle un fichero para que lo interprete, que contenga desde texto plano hasta etiquetas de algún lenguaje de marcado (por ejemplo SABLE):

festival example.sable --tts

Este método, aunque pueda ser más cómodo, es ineficiente en caso de querer hacer varias reproducciones consecutivas, pues de forma transparente al usuario para cada instrucción Festival debe iniciarse, cargar su configuración, ejecutar la instrucción recibida, y cerrarse liberando recursos.

Hablemos ahora de un aspecto muy importante: El juego de caracteres[53] o charset. Es posible que realizando alguna prueba con Festival, hayamos escuchado algún sonido extraño en lugar de lo que simplemente era algo como una vocal acentuada o una eñe. Esto se debe a que la mayoría de los sistemas operativos codifican las cadenas de texto en UTF-8 (Formato de Transformación Unicode de 8 bits), un estándar pensado para las lenguas anglosajonas que no contempla los caracteres especiales de otros idiomas como la "ç" o nuestras citadas tildes o "ñ". Es por ello que deberíamos hacer una preconversión de este juego de caracteres al adecuado para nuestro castellano: el ISO-8859-1. Para ello utilizaremos un programa llamado "iconv"[54]:

La versión 1.7-2 de Iconv estaba disponible en nuestra instalación por defecto de Ubuntu (versión 10.10), y es probable que nuevas versiones sigan disponibles en sucesivas instalaciones. En cualquier caso, podríamos obtenerlo de la forma habitual escribiendo la siguiente línea en consola:

apt-get install libtext-iconv-perl

Hagamos una prueba para ver la diferencia de reproducir un texto con caracteres propios de nuestra codificación, primero sin convertir y luego con conversión a nuestro charset ISO-8859-1:

Sin convertir (UTF-8):

echo "muchísimas castañas" | festival --tts --language spanish

Habremos apreciado que la síntesis realizada no se parece en nada a lo esperado; en lugar de leer "muchísimas castañas", ha deletreado las palabras y ha dicho "equis equis" cuando se ha encontrado una tilde y una eñe. Este "equis equis" se debe a que ambos caracteres se han traducido como dos símbolos extraños cada uno en este charset, y la voz castellana principal nos alerta de

su presencia nombrándolos como si fueran la letra “x” (las voces de la Junta de Andalucía dicen “ups” en lugar de “equis”).

Ahora realizamos la conversión previa a ISO-8859-1:

echo "muchísimas castañas" | iconv -f utf-8 -t iso-8859-1 | festival --tts --language spanish

Ahora sí que suena como esperábamos. Lo que hemos hecho ha sido direccionar la salida del “echo” hacia el “iconv”, le hemos indicado que convierta desde (-f, from) UTF-8 a (-t, to) ISO-8859-1, y hemos enviado el resultado a la entrada de Festival, que ha dispuesto de unos caracteres totalmente reconocibles en nuestro idioma. Estas conversiones serán, por tanto, vitales en nuestras aplicaciones en castellano.

Para terminar, Festival incluye una utilidad que nos permite guardar en un fichero de audio (.wav) el resultado de un texto o archivo sintetizado, en lugar de reproducirlo por la salida estándar. Se llama Text2wave y para utlizarlo escribiremos desde consola (no desde el intérprete) una línea del siguiente estilo:

text2wave texto.txt -otype snd -o habla.wav

En la carpeta “EJEMPLOS/SINTESIS TTS” hay disponibles dos ficheros: “Memoria”, que no es más que un fichero de texto plano con los dos primeros capítulos de la presente memoria, y “memoria.wav”, que es el fichero de audio creado con *text2wave* a partir del procesamiento realizado por nuestra aplicación e-Narrador (sección 5.1) sobre el fichero citado.

Pero todo esto aparte, la manipulación de Festival que más nos interesa será desde su API[55] C++ que nos permitirá controlarlo desde las aplicaciones, como mostramos en la siguiente sección.

3.4_ API C++

Gracias al paquete festival-dev que hubimos instalado, tenemos a nuestra disposición ciertas cabeceras y funciones que nos permitirán integrar y manejar de una forma potente el motor de síntesis de Festival. Para ello, al principio de nuestros programas deberemos incluir la librería de festival:

#include<festival.h>

A continuacón, en la función principal *main*, deberemos inicializar Festival. La primera de las tres líneas establece el tamaño de la pila para las instrucciones de Festival, la segunda indica que queremos que se cargue el fichero de configuración de Festival (1: cargarlo, 0: ignorarlo), y la tercera es una función que lanza estas dos preferencias:

```
int heap_size = 210000;
int load_init_files = 1;
festival_initialize(load_init_files,heap_size);
```

A partir de este punto tenemos libertad para utilizar cualquiera de estas otras seis instrucciones de Festival como si de cualquier típica instrucción de C++ se tratara:

festival_say_file(const EST_String &filename)

Lee el contenido del fichero suministrado, ya sea texto plano o formateado con etiquetas de un lenguaje de marcado.

festival_say_text(const EST_String &text)

Lee el texto suministrado en la cadena argumento. Es el equivalente a la orden *(SayTexy "XXXXX")* del intérprete.

festival_eval_command(const EST_String &expr)

Interpreta la cadena pasada como argumento como una orden del intérprete de Festival. Puede usarse, entre otras cosas, para ejecutar la orden que cambia la voz utilizada. Por ejemplo:
festival_eval_command("(voice_JuntaDeAndalucia_es_pa_diphone)")
cambiaría la voz actual a la masculina de la Junta de Andalucía.

festival_load_file(const EST_String &filename)

Carga el fichero .scm (Scheme) indicado que contenga líneas de órdenes de Festival y las ejecuta secuencialmente. Es el equivalente a la orden *(load "XXXXX.scm")* del intérprete.

festival_text_to_wave(const EST_String &text,EST_Wave &wave)

Sintetiza el texto suministrado como argumento y lo guarda en un archivo de audio en lugar de reproducirlo. Es el equivalente a utilizar el programa text2wave desde consola.

festival_wait_for_spooler()

La instrucción *Festival_Say_File* pone el sistema en modo asíncrono, por lo que necesitamos esta instrucción para indicar que queremos esperar a que se termine de leer el último fichero antes de proseguir la ejecución del programa.

Vamos a escribir y a compilar un ejemplo simple, disponible en la carpeta "EJEMPLOS/API FESTIVAL C++" del anexo. Lo escribiremos en un fichero que llamaremos "ejemplo.cc", y contendrá las siguientes líneas:

```
#include <stdio.h>
#include <festival.h>

int main(int argc, char **argv)
{
     int heap_size = 210000;
     int load_init_files = 1;

     festival_initialize(load_init_files, heap_size);
     festival_eval_command("(voice_el_diphone)");
     festival_say_text("Hola mundo");
}
```

La orden para compilar cualquier programa que utilice Festival deberá tener esta forma, enlazando con los ficheros necesarios:

```
g++ ejemplo.cc -I/usr/lib -l Festival -I/usr/include/festival -I/usr/include/speech_tools/ -leststring -lestbase -lestools -o ejemplo
```

En la carpeta del anexo mencionada hay disponible un Makefile con esta orden, lo cual nos permite realizar la compilación instroduciendo simplemente la orden "make". Si todo ha ido bien, tendremos el fichero "ejemplo" listo para ejecutar, tras lo que escucharemos las palabras "hola munda".

Ahora ya tenemos suficientes conocimientos para ponernos manos a la obra realizando nuestras propias aplicaciones en C++ aprovechando las características de Festival.

4_ SABLE

4.1_ ¿QUÉ ES?

SABLE es un lenguaje de marcado de texto, que nace con la intención de convertirse en un estándar de los ficheros de representación de la síntesis TTS. Es el heredero de otros dos estándares que vimos en la sección 2.2.1: el STML y el SSML (que a su vez era la evolución del JSML), y mantiene la misma estructura de etiquetas XML. Al igual que los anteriores, ha sido desarrollado por el consorcio W3C.

Un fichero SABLE es, en esencia, un fichero de texto con extensión .sable que contiene las etiquetas correspondientes a dicho lenguaje acotando fragmentos de texto con propósitos diversos. Este lenguaje, totalmente compatible con Festival, nos permite controlar aspectos de la pronunciación y la entonación de la lectura, así como identificación de fragmentos que deben leerse de forma diferente a la habitual (fechas, teléfonos, números ordinales, etc.). En definitiva, mecanismos que nos permiten humanizar en lo posible un proceso de síntesis que da como resultado, por sí solo, una lectura plana e impersonal.

4.2_ PRESTACIONES

La sintaxis de SABLE[56] está compuesta por el siguiente conjunto de etiquetas:

<MARK>
Establece una marca que los motores de síntesis pueden utilizar para enviar señales de por ejemplo zona alcanzada.

<EMPH>
Atributos: LEVEL={0.0, 0.5, 1.0, 1.5}
Modifica el nivel de énfasis de la voz.

<BREAK>
Atributos: LEVEL={0.0, 1.0, 2.0, 3.0}
MSEC=[float_milisegundos]
TYPE={? ! . ,}
Hace una pausa en la lectura, pudiendo especificar la duración y el tipo de pausa (interrogativa, exclamativa, punto, coma).

<PITCH>
Atributos: BASE=[número %] ó
{default, lowest, low, medium, high, highest}
MIDDLE=[número %] ó

{default, lowest, low, medium, high, highest}
RANGE=[número %] ó
{default, smallest, small, medium, large, largest}
Permite modificar el tono de la voz.

<RATE>
Atributos: SPEED=[número %] ó
{slowest, slow, medium, fast, fastest}
Modifica la velocidad de la lectura.

<VOLUME>
Atributos: LEVEL=[número %] ó
{quiet, medium, loud, loudest}
Modifica el volumen de la voz.

<AUDIO>
Atributos: SRC=[dirección_fichero_sonido]
MODE={background, insertion}
LEVEL=[float_entre_0_a_1]
Reproduce el archivo de sonido especificado en SRC, indicando si se debe reproducir en solitario (insertion) o a la vez que el resto del texto (background), y a qué volumen.

<ENGINE>
Atributos: ID=[identificador_motor_de_síntesis]
DATA=[cadena_a_sustituir]
Sustituye el texto indicado cuando el motor de síntesis utilizado es el especificado.

<MARKER>
Atributos: MARK=[cadena_identificadora]
Sirve como punto de anclaje para marcas.

<SABLE>
Identifica el documento actual como un documento SABLE.

<PRON>
Atributos: IPA=[cadena_descripción_pronunciación]
SUB=[cadena_pronunciación_deseada]
ORIGIN=[cadena_lenguaje_origen]
Especifica una pronunciación diferente a la literal para cierto texto.

<SAYAS>
Atributos: MODE={name, cardinal, ordinal, measure, fraction, math, currency, postal, net, phone, time, date, literal}
MODETYPE={URL, EMAIL, HMS, HM, MD, MY, YM, YMD, MDY, DMY}
Indica una manera especial de leer el texto indicado (fechas, horas, teléfonos, fracciones, etc).

<LANGUAGE>
Atributos: ID=[identificador_lenguaje]
CODE=[identificador_opcional_lenguaje]
Especifica el lenguaje en el que leer el texto contenido, según su identificador en la tabla ISO-639:
http://xml.coverpages.org/iso639a.html

<SPEAKER>
Atributos: GENDER={male, female}
AGE={child, teen, younger, middle, older}
NAME=[nombre_voz]
Especifica la voz con la que leer el texto contenido.

<DIV>
Atributos: TYPE={paragraph, sentence}
Estructura el texto en frases y párrafos.

En la carpeta "EJEMPLOS/SABLE" del anexo hay disponible un fichero .sable (example.sable), fragmento de un ejemplo[57] confeccionado por Allan W. Black, profesor de la Facultad de Informática de la Carnegie Mellon University. En él podemos apreciar el resultado de ejecutar muchas de las etiquetas que hemos repasado. Para ello, recordemos que basta introducir en consola la siguiente línea:

festival example.sable –tts

Sin embargo, SABLE no se interpreta correctamente en otros idiomas que hemos probado que no sean aquellos originales del proyecto, en inglés. Esto se puede deber a que la creación de las demás voces no se haya hecho siguiendo a rajatabla las especificaciones, probablemente debido a que ello conlleve un considerable aumento de la complejidad en la creación de éstas.

En la carpeta "EJEMPLOS/SABLE/COMPARATIVA" hay disponibles tres ficheros SABLE correspondientes a tres demostraciones usando este lenguaje para pequeños fragmentos en inglés, español e italiano. No es necesario instalar la voz italiana para comparar los resultados, pues están guardados los resultados de la síntesis en sus correspondientes ficheros de audio, pero si se desea se puede instalar con la siguiente orden:

apt-get install festvox-itapc16k

Tras escucharlos es fácil comprobar que, como ya avanzábamos, sólo la lectura en inglés ha interpretado las etiquetas de SABLE, ignorándolas por completo en español y en italiano. Esta es

precisamente la razón por la que hemos creado una versión inglesa del lector de noticias, ya que era el único idioma que entendemos en el que podíamos hacer una demostración utilizando SABLE.

4.3_ DE XML A SABLE → XSL

Actualmente, la gran mayoría de webs de prensa digital tienen a disposición de los usuarios lo que se denomina “sindicación RSS”[58]. Estos RSS, en la práctica, no son más que ficheros basados en una estructura de etiquetas XML que delimita y agrupa la información textual contenida en ellos (títulos de noticias, cuerpos de noticias, hipervínculos, etc). La utilización principal de estos ficheros es automatizar la obtención de las noticias en aplicaciones de los usuarios o su publicación en sus webs o blogs, pero nosotros aprovecharemos esa estructura XML para algo más allá. Puesto que los ficheros SABLE comparten ese mismo tipo de estructura XML, es posible realizar una transformación directa de RSS a SABLE mediante lo que se denominan hojas XSL.

XSL[59] (Extensible Stylesheet Language, lenguaje extensible de hojas de estilo) es un lenguaje basado en el estándar XML que se utiliza para especificar cómo la información contenida en un fichero de tipo XML debe ser transformada o formateada para su presentación en otro fichero o medio. Consta de varias etiquetas que nos permiten básicamente seleccionar qué información queremos extraer, qué debemos ignorar, y qué otro texto indicado explícitamente queremos que aparezca, como es el caso de las cabeceras y las etiquetas html o SABLE. Por ejemplo, dado el siguiente contenido de un fichero XML:

```
<?xml version="1.0" ?>
<tienda>
     <producto>
          <nombre>arroz</nombre>
          <precio>0,50</precio>
     </producto>
     <producto>
          <nombre>azúcar</nombre>
          <precio>0,30</precio>
     </producto>
     <producto>
          <nombre>sal</nombre>
          <precio>0,18</precio>
     </producto>
</tienda>
```

Y el siguiente fichero XSL que selecciona el contenido de las etiquetas <nombre> y <precio> de cada <producto>:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
<xsl:output method="text"/>

<xsl:template match="/">
<xsl:apply-templates select="/tienda"/>
</xsl:template>

<xsl:template match="producto">
<xsl:value-of select="nombre"/>: <xsl:value-of select="precio"/> €.
</xsl:template>

</xsl:stylesheet>
```

Al aplicar la transformación obtenemos lo siguiente:

arroz: 0,50 €.

azucar: 0,30 €.

sal: 0,18 €.

Describamos los pasos del fichero XSL. Primero, lo que se ha hecho es indicar qué tipo de archivo es, un XSL, y también se ha indicado que la salida queremos que sea un fichero de texto plano. Después, se ha indicado que queremos buscar en las etiquetas que haya anidadas dentro de <tienda> (es decir, entre la apertura <tienda> y el cierre </tienda>). Las siguientes líneas indican que de todo producto copiaremos el contenido de las etiquetas <nombre>, tras lo que pondremos dos puntos (:), después copiaremos el contenido de las etiquetas <precio>, añadiremos un espacio y el símbolo “€”, y acabaremos poniendo un punto. Sólo queda cerrar la etiqueta de la hoja de estilos y ya está lista la hoja de transformación XSL.

Para aplicar las transformaciones se necesita usar un programa que las procese, como por ejemplo Xalan[60], procesador libre creado por la Apache Software Foundation. Si queremos probar lo que hemos visto, podemos instalarlo de la siguiente manera:

apt-get install xalan

En la carpeta “EJEMPLOS/XSL” del anexo están disponibles los ficheros tienda.xml y extrae.xsl del ejemplo que hemos comentado, así como la salida que se produce al aplicar la transformación con Xalan, que si hemos instalado correctamente el programa se hará así:

xalan -in tienda.xml -xsl extrae.xsl -out salida.txt

De manera similar será como procederemos para obtener un fichero .sable a partir de un .xml, aprovechando que podemos incluir explícitamente el texto que deseemos. Así, del mismo modo que en el ejemplo añadíamos el símbolo “€” al resultado final, podemos añadir todas las etiquetas y cabeceras de SABLE para conseguir un fichero válido de este tipo. Esto es lo que haremos en la versión en inglés de nuestro lector de noticias (Web2Voice), que aunque no utilizará el procesador Xalan que quizá acabamos de instalar, sí usará las librerías para C++ de éste que nos permitirán realizar las transformaciones de manera casi idéntica desde la aplicación, y que en su momento instalaremos.

5_ APLICACIONES DESARROLLADAS

5.1_ E-NARRADOR

5.1.1_ PROPÓSITO

El objetivo de esta aplicación es disponer de un lector de libros o documentos digitales, proporcionando la posibilidad de continuar la lectura por donde se dejó la última vez, para permitir así la lectura fraccionada de volúmenes de cualquier extensión mediante una marca equivalente a la señal que colocamos en los libros de texto convencionales.

Todas las opciones y mensajes que aparezcan en el programa se presentarán de forma auditiva, lo cual posibilita el acceso a las personas invidentes o de visión reducida, pero también de forma escrita, lo cual refuerza y facilita el entendimiento al resto.

Aunque temporalmente esta aplicación fue la última en ser desarrollada, la presentamos en primer lugar por ser la más simple en cuanto a programación se refiere.

5.1.2_ PRERREQUISITOS

Para la correcta compilación y ejecución de la aplicación, se deberán instalar los siguientes programas y librerías. Nombraremos todos los paquetes y programas necesarios sin tener en cuenta si ya se han instalado préviamente para probar los ejemplos de secciones anteriores o para hacer funcionar alguna de las otras dos aplicaciones presentadas.

Festival, su API de programación, y voz en castellano
El motor de síntesis del habla Festival, las cabeceras y funciones para su integración en aplicaciones, y la voz oficial en castellano.
Versión utilizada: 1:2.0.95-beta-2ubuntu1
Orden de consola para instalación desde el repositorio:

apt-get install festival festival-dev festvox-ellpc11k

Iconv
El conversor entre juegos de caracteres desde el shell. En nuestra distribución 10.10 de Ubuntu ya se encontraba instalado.
Versión utilizada: 1.7-2
Orden de consola para instalación desde el repositorio:

apt-get install libtext-iconv-perl

Lo ideal habría sido utilizar las librerías para C++ del *iconv* en lugar de depender de la ejecución externa desde consola mediante la orden *system*. Esto lo propondremos en Trabajos Futuros (sección 7).

Voces de la Junta de Andalucía
Las voces en castellano de considerable calidad de la Junta de Andalucía, una masculina y otra femenina.
Versión utilizada: 1.0-1
Instalación no disponible desde el repositorio. Ver sección 3.2, página 13.

G++
El compilador GNU de C++.
Versión utilizada: 4:4.4.4-1ubuntu2
Orden de consola para instalación desde el repositorio:

apt-get install g++

Por último recordar que el Sistema Operativo en el que se ha creado y probado la aplicación es:
Ubuntu
Versión: 10.10

En caso de disponer de versiones de software o del sistema operativo distintos a los recién indicados, no podemos asegurar que el funcionamiento sea idéntico al descrito en la siguiente subsección.

5.1.3_ FUNCIONAMIENTO

La aplicación completa está disponible en la carpeta "APLICACIONES/eNarrador" del anexo.

En primer lugar compilamos la aplicación. Para ello, basta con situarnos en el directorio correspondiente a ésta desde la consola y ejecutar la herramienta de generación de código introduciendo "make", aprovechando que hay incluído un Makefile que contiene la siguiente línea necesaria para la compilación:

g++ enarrador.cc -I/usr/lib -l Festival
-I/usr/include/festival -I/usr/include/speech_tools/
-leststring -lestbase -lestools -o enarrador

Si todo ha ido bien, tras esto ya tendremos el ejecutable “enarrador” listo en el directorio. Hagamos un repaso por el contenido inicial de la carpeta del programa.

Carpetas:

Archiaux: Contendrá ficheros auxiliares correspondientes a los refinamientos y conversiones de charsets del documento a leer.

Textos: En esta carpeta será donde debemos situar los documentos, en formato de texto plano, que queremos tener disponibles para lectura cuando iniciemos el programa. Como ejemplo contiene inicialmente cuatro cuentos de Edgar Allan Poe: El corazón delator, El gato negro, El retrato oval, y La máscara de la muerte roja.

Guías: Aquí es donde el programa creará y mantendrá actualizados ficheros con el mismo nombre que los documentos leídos más la extensión “.guia”, que guardarán el número de párrafo al que volver si se desea continuar la lectura en una ejecución posterior, siendo eliminados tras finalizar la lectura del documento completo.

Ficheros:

enarrador: Ejecutable de la aplicación.

enarrador.cc: Fichero fuente de la aplicación. En la siguiente sección explicaremos cada una de las funciones que contiene.

Makefile: Fichero con la orden para generar la aplicación a través de la herramienta de generación “make”.

sustituciones: fichero que contiene especificaciones de pronunciación alternativa a la literal para una serie de palabras, que el propio usuario puede ampliar con sus propias experiencias. Lo veremos mejor un poco más adelante en esta misma sección.

Funcionamiento paso a paso:

En primer lugar ejecutamos el programa escribiendo “./enarrador” en la consola. Si es la primera vez que lo ejecutamos, un mensaje nos alertará de esto y se procederá a buscar y almacenar el controlador de sonido adecuado. A continuación, se realizará la comprobación de las voces instaladas, que en caso de que falte alguna de las que se utilizarán, avisará de que la experiencia auditiva puede ser de inferior calidad a la esperada.

En cualquier caso, a continuación pueden darse tres casos:
1- Que la carpeta “Textos” esté vacía y no haya nada disponible para leer, con lo cual tras indicárnoslo con un aviso se procede a cerrar la aplicación.
2- Que en la carpeta “Textos” haya un sólo fichero, que automáticamente se considerará el fichero a leer.
3- Que en la carpeta “Textos” haya al menos dos ficheros, tras lo cual un menú nos pide que pulsemos la barra espaciadora al escuchar el nombre del fichero que deseamos, pues irá nombrando uno a uno todos ellos. Al hacerlo, ese fichero será el seleccionado como fichero a leer.

Una vez está designado el fichero pueden presentarse otros dos casos:
1- Que sea la primera vez que se lee el fichero o se hubiera terminado de leer por completo la última vez que se leyó, tras lo cual la aplicación estará dispuesta para empezar a leer el fichero desde el principio.
2- Que el fichero se hubiera dejado a medias la última vez que se leyó, en cuyo caso se nos presentarán dos opciones: Si pulsamos la barra espaciadora, se preparará para continuar la lectura por donde se dejó. Si por el contrario pulsamos Intro, se dispondrá a leer desde el principio, borrando la marca de lectura que había guardada.

A partir de aquí, la aplicación comenzará a leer el fichero desde la línea correspondiente, recordándonos que para salir al final del párrafo que se esté leyendo en un momento dado y guardar la posición de lectura, se debe pulsar Escape. Si efectivamente finalizamos con Escape la lectura, la aplicación guarda la posición, se despide y se cierra. Si dejamos que acabe la lectura hasta el final, se vuelve al menú de selección de documento.

Descripción de las funciones

Vamos a describir el funcionamiento de cada una de las funciones implicadas en la aplicación, las cuales están disponibles al completo en la siguiente sección (5.1.4).

(1) → int Main(int argc, char **argv)

Es la función principal, el punto de entrada de nuestra aplicación. En ella nos limitamos a hacer secuencialmente:
1- Inicializar Festival.
2- Llamar a la función **InicializarControlador→(2)**.
3- Llamar a la función **ComprobarVoces→(5)**.
4- Llamar a la función **ListarCarpeta→(6)**.

(2) → void InicializarControlador()

Es la función encargada de cargar el controlador de sonido adecuado para nuestro equipo. Lo que hace es comprobar si existe un fichero llamado "configaudio" en el directorio principal de la aplicación. Si no existe, llama a la función **BuscaControlador→(3)**, que devuelve un código que si es distinto de cero se almacena en el fichero "configaudio" que crea en ese mismo instante, mientras que si es cero se alerta de que no se encuentra un controlador adecuado y se finaliza la aplicación. Después utiliza ese código (entre 1 y 6) para ejecutar una instrucción de Festival que cambia el controlador de sonido a utilizar correspondiente a dicho código. Finalmente limpia la pantalla.

(3) → char BuscaControlador()
-Devuelve: el código del controlador adecuado entre 1 y 6, o un 0 si no encuentra ninguno.

Esta función no hace otra cosa que ir cargando uno a uno los seis controladores de sonido que conocemos, intentando reproducir para cada uno de ellos una frase afirmando que se ha encontrado dicho controlador, utilizando para realizar la lectura la función **Decir→(4)**. La aplicación cronometra esos intentos de lectura de cada frase y comprueba el tiempo en segundos obtenido; Sólo cuando ese tiempo sea igual o mayor a dos segundos, significará que la frase efectivamente se ha leído, por lo que se considera válido el último controlador cargado y se devuelve su código correspondiente. Si ningún controlador ha sido válido, el valor de retorno será entonces cero.

Hemos tenido que idear una estrategia de detección así de curiosa, porque las funciones de Festival no se han comportado como esperábamos: la función que cambia de controlador devuelve 1 siempre independientemente de si el controlador es válido o no, y la función para leer un texto devuelve 1 tanto si consigue leer como si no.

(4) → void Decir(string eltexto)
-Recibe: La cadena de texto a leer.

El objetivo de esta función es convertir la cadena de texto suministada en un fichero con codificación ISO-8859-1 para evitar los errores de lectura de algunas palabras castellanas con la codificación UTF-8, como comentábamos en la sección 3.3.

En primer lugar se crea un fichero "archiaux" en la carpeta "Archiaux" y se guarda en él la cadena suministrada a la función como argumento. A continuación se hace una conversión de dicho fichero, codificado en UTF-8, a otro llamado "archiauxISO8859" codificado en ISO-8859-1, borrando inmediatamente después el

primero. Después se extrae la línea almacenada en el nuevo fichero, se cambia la voz a utilizar a la castellana principal, y se lee dicha línea, tras lo que se borra también el último archivo.

Como se puede ver, las conversiones entre juegos de caracteres se realizan invocando la orden iconv desde consola, utilizando para ellos la instrucción system. Como ya comentábamos en los prerrequisitos, sería preferible utilizar en su lugar las librerias de C++ (ver Trabajos Futuros, sección 7).

(5) → void ComprobarVoces()

En primer lugar, se almacenan en dos arrays las rutas completas de las carpetas donde deberían estar las voces requeridas por la aplicación, y también únicamente el nombre en otros dos. Después sólo hay que intentar abrir los directorios correspondientes a dichas rutas e indicar si el intento ha sido fallido con alguno, lo que significará que no ha sido instalado, y avisará de la sensible pérdida de calidad de la experiencia debido a ello.

(6) → void ListarCarpeta()

Desde aquí es desde donde comienza la parte específica de la aplicación. Lo primero que hace es contar cuántos archivos hay en la carpeta "Textos" mediante la función *readdir*, contenida en la librería *dirent.h*, que permite un recorrido secuencial por todas las entradas del directorio indicado. Como también cuenta la entrada del directorio superior (..) y la del actual (.), restamos 2 al resultado. Entonces, según ese número de archivos sea uno u otro, se toma una acción distinta:
-Si es 0, se avisa que no hay archivos disponibles, y se finaliza la ejecución de la aplicación.
-Si es 1, se llama a la función **ObtenerArchivo→(7)** para obtener el nombre de ese fichero y se le pasa a la función **SeleccionarDocumento→(9)**.
-Si es 2 o mayor, un menú va leyendo los nombres de cada uno de los ficheros, consultándolos mediante **ObtenerArchivo→(7)**, hasta que la comprobación de **ComprobarTeclado→(8)** sobre si se ha pulsado Espacio devuelve True (limpiando la entrada estándar entre comprobación y comprobación mediante **VaciarBufferTeclado→(10)**), con lo cual se llama a **SeleccionarDocumento→(11)** con el fichero correspondiente como argumento.

(7) → string ObtenerArchivo(int numarch)
-Recibe: el número de entrada de archivo a consultar.
-Devuelve: el nombre del archivo correspondiente al número.

Recorre una a una las entradas del directorio “Textos” y devuelve el nombre del archivo cuya numeración coincida con la suministrada como argumento. Como puede verse, se ignoran las entradas de directorio actual (.) y directorio superior (..).

(8) → char ComprobarTeclado()
-Devuelve: la última tecla pulsada cuando ésta sea distinta de Escape.

Esta función agrupa la obtención de la última tecla pulsada desde la última limpieza del buffer, con **UltimaTeclaPulsada→(9)**, y la salida del programa cuando esta tecla sea Escape. Esta agrupación evita un exceso de comprobaciones en los menús.

(9) → char UltimaTeclaPulsada()
-Devuelve: la última tecla pulsada.

Esta función es una versión modificada de una versión de la función C++ de Borland *Kbhit*, propuesta por el usuario *itsme86* en el foro *cprogramming*. La función en concreto se encuentra en el siguiente enlace:
http://cboard.cprogramming.com/c-programming/63166-kbhit-linux.html

La modificación realizada al final de la función permite que en lugar de saber únicamente si se ha pulsado una tecla cualquiera o no, sepamos la tecla exacta.

(10) → void VaciarBufferTeclado()

Esta escueta función únicamente realiza dos acciones: la línea superior sirve para deshabilitar la entrada estándar, lo cual a su vez vacía el buffer correspondiente. La segunda línea vuelve a habilitarla. La función es necesaria porque la comprobación de teclado realizada por las funciones del teclado no vacía el buffer.

(11) → void SeleccionarDocumento(string elnombredoc)
-Recibe: el nombre del documento a leer.

En primer lugar se preparan las cadenas con las rutas, tanto del fichero a leer como del fichero que almacenará la posición de lectura (que no es más que el nombre del fichero añadiendo “.guia” al final, y en la carpeta “Guias” en lugar de “Textos”). Después se intenta abrir ese fichero “.guia”. Si existe, significa que hay una posición de lectura de dicho texto almacenada de una lectura anterior, la cuál un menú da la oportunidad de recuperar. Se

comprueba el teclado con **ComprobarTeclado→(8)**, obteniendo el número de línea por la que continuar el texto cuando la tecla es Espacio, o manteniendo el número de linea igual a cero cuando la tecla es Intro. Para finalizar, se pasa a la función **LeerDocumento→(12)**.

(12) → void LeerDocumento(FILE * eldoc, int lalinea, char archiguia[512])
-Recibe: el descriptor del fichero a leer, la línea por la que continuar la lectura y la ruta del fichero ".guia" correspondiente al fichero a leer.

Lo primero que se hace es crear un fichero llamado "leyendo.txt" en la carpeta "Archiaux", en el que únicamente se copia el contenido del fichero a leer a partir de la línea recibida como argumento. A continuación la función **RefinarTexto→(13)** crea otro fichero a partir de ese "leyendo.txt" llamado "leyendoRef.txt" dentro de la misma carpeta "Archiaux". Ese "leyendoRef.txt" se convierte de codificación UTF-8 a ISO-8859-1 para que sea leído en castellano correctamente, resultando otro fichero llamado "leyendoISO8859.txt" dentro del mismo directorio, que será el definitivo para leer. Entonces, tras un aviso de que se va a proceder a leer el documento, se intenta cargar la voz masculina de la Junta de Andalucía y se empiezan a obtener líneas una a una para ser leídas. Mientras no se llegue al final del fichero, después de leer cada línea se suma uno a la línea actual y se guarda ese número en el fichero ".guia" correspondiente. Si se llega al final del fichero, se borra el archivo ".guia" y si en cambio se sale pulsando Escape con la comprobación habitual se abandona la aplicación.

(13) → void RefinarTexto()

Esta función tiene como objetivo corregir y mejorar el documento a leer por Festival. Debido a su complejidad, el propio código contiene comentarios explicativos de las comprobaciones y sustituciones que se realizan en cada línea. Probablemente realiza más comprobaciones de las necesarias para esta aplicación, ya que es una función reutilizada en común con la aplicación de lectura de noticias "Web-a-Voz".

Básicamente, el puntero de lectura del fichero origen ("leyendo.txt") va copiando al fichero destino ("leyendoRef.txt") los caracteres que encuentra secuencialmente, excepto en las comprobaciones que va realizando, como al encontrar caracteres extraños, paréntesis, espacios, signos de puntuación excesivos, etc. En definitiva cualquier carácter que tras realizar pruebas establecimos que Festival ignora o no pronuncia adecuadamente. Algunas comprobaciones necesitan consultar los caracteres que

aparecerán más adelante con **ProxCaracteres→(14)**, caracteres anteriores con **AntCaracteres→(15),** u obtener cuántas cifras consecutivas aparecen con **ComprobarSigNumeros→(16)** para establecer si por ejemplo se debe mantener el punto separador de miles. Además, se comprobará si en el texto hay presentes palabras que contemplamos en el fichero "sustituciones", con **EncontrarSustitucion→(17)**.

(14) → bool ProxCaracteres(FILE * fdesc, string laetiqueta)
-Recibe: el descriptor del fichero de lectura y el texto a comprobar.
-Devuelve: 'True' cuando la cadena recibida se encuentra a partir de la posición actual del cabezal de lectura en el fichero, o 'False' en caso contrario.

Incialmente se guarda la posición del puntero de lectura del fichero, y se procede a comparar los siguientes caracteres uno a uno con los del string suministrado. Si alguno de ellos no coincide, se devuelve 'False' y se devuelve el puntero de lectura a su posición inicial, mientras que si todas las comparaciones tienen éxito se devuelve 'True'.

(15) → bool AntCaracteres(FILE * fdesc, string laetiqueta)
-Recibe: el descriptor del fichero de lectura y el texto a comprobar.
-Devuelve: 'True' cuando la cadena recibida se encuentra a partir de la posición actual del cabezal de lectura del fichero, o 'False' en caso contrario.

De forma análoga a la función anterior, ésta almacena la posición actual del puntero de lectura y procede a comparar carácter a carácter, pero esta vez hacia atrás. Si alguna comparación falla devuelve 'False' y si todas tienen éxito devuelve 'True'. En cualquiera de los dos casos devuelve el puntero de lectura a la posición almacenada inicialmente.

(16) → int ComprobarSigNumeros(FILE * fdesc)
-Recibe: el descriptor del fichero de lectura.
-Devuelve: el número de caracteres numéricos existentes desde la posición actual.

Muy parecida a las dos recién descritas, esta función guarda la posición actual del puntero de lectura del fichero, y procede a analizar las siguientes posiciones hasta que encuentre un carácter que no sea una cifra, manteniendo el recuento de éstas. Finalmente se devuelve el puntero de lectura a la posición guardada inicialmente y se devuelve el número de cifras consecutivas.

(17) → bool EncontrarSustitucion(FILE * orig, FILE * dest, FILE * dicci, char ultchar)
-Recibe: el descriptor del fichero de origen, el descriptor del fichero destino, el descriptor del fichero de sustituciones y el último carácter encontrado en el fichero de origen.
-Devuelve: Si se ha aplicado alguna sustitución ('True') o no ('False').

Esta función también la utilizamos originalmente en la aplicación de lectura de noticias web (Web-a-Voz), pero podemos aprovechar su potencial para cualquiera aplicación TTS que se nos ocurra. Nos permite utilizar un fichero llamado "sustituciones", ampliable por cada usuario según sus experiencias, que sustituye palabras que leídas literalmente no suenan como deberían por una escritura lo más aproximada a cómo deberían pronunciarse, como es el caso de abreviaturas, siglas o nombres extranjeros. Por ejemplo, si el lector encontrara la palabra "Michael" y la leyera normalmente, la pronunciación sería tal cual "mi-cha-el" como si de una palabra española más se tratara, pero con una entrada de sustitución adecuada en el fichero de la siguiente forma: "michael>máiquel" le indicamos que cuando encuentre esa palabra no debe leerla así, sino como se especifica a la derecha del carácter ">". En la carpeta "EJEMPLOS/SUSTITUCIONES" del anexo hay disponibles algunas noticias ficticias creadas explícitamente para contener varias de las palabras contempladas en el fichero de sustituciones, junto con dos archivos de audio, resultado de leerlas utilizando dicho fichero ("CON_sustituciones.wav") y sin utilizarlo ("SIN_sustituciones.wav"). Como se puede apreciar, la diferencia es más que notable. Veamos cómo funciona:

En primer lugar, comprueba si el último carácter leído no era una letra, con lo cual ya se sabe que no se aplicará una sustitución y se devuelve 'False'. En otro caso, se almacenan los siguientes caracteres del fichero origen, y se van comparando con las entradas del fichero de sustituciones línea a línea. Para cada una de las líneas se hace una comparación carácter a carácter, hasta que una comparación resulta fallida, lo cual hace que se pase a la siguiente línea, o hasta encontrar el carácter ">" en el fichero de sustituciones y un espacio, punto, coma, paréntesis o cambio de línea en la misma posición en el fichero de origen, lo cual significa que la palabra comprobada tiene una sustitución contemplada. En este último caso se procede a escribir en el fichero destino lo que haya a continuación del ">" en el fichero de sustituciones, se sitúa el puntero de posición justo después de la palabra sustituída y se devuelve 'True'. Si ninguna sustitución ha sido posible después de comprobar todas las líneas, se devuelve 'False'.

5.1.4_ CÓDIGO FUENTE

A continuación se expone el código fuente de la aplicación “e-Narrador” al completo:

```
#include <stdio.h>
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <festival.h>
#include <sstream>
#include <string>
#include <termios.h>
#include <fcntl.h>
#include <dirent.h>

void ListarCarpeta(), InicializarControlador(), ComprobarVoces(), RefinarTexto(),
Decir(string), VaciarBufferTeclado();
void SeleccionarDocumento(string), LeerDocumento(FILE*,int,char*);
bool ProxCaracteres(FILE*,string), AntCaracteres(FILE*,string),
EncontrarSustitucion(FILE*,FILE*,FILE*,char);
int ComprobarSigNumeros(FILE*);
char BuscaControlador(), UltimaTeclaPulsada(), ComprobarTeclado();
string ObtenerArchivo(int);

int heap_size = 210000;
int load_init_files = 1;

int main(int argc, char **argv)
{
    bool salir = false, haleido = false;
    char opcionchar;
    int opcionint;

    festival_initialize(load_init_files,heap_size); //se inicializa Festival

    InicializarControlador();    //se busca el controlador de sonido adecuado

    ComprobarVoces();            //se comprueba si están instaladas las voces utilizadas

    ListarCarpeta();             //se da a elegir el fichero para leer de entre los
disponibles en le carpeta 'Textos'
}

//---------------------------------------------------------------------
//Función para mostrar y elegir los textos disponibles para lectura.
//---------------------------------------------------------------------
void ListarCarpeta()
{
    struct dirent *d;
    const char *direccion = "./Textos";
    char *nomdocu;
    string docu;
    int numtextos = -2;

    DIR *directorio = opendir(direccion);
    while ((d=readdir(directorio)) != NULL)   //contamos cuántos ficheros hay en la
carpeta 'Textos'
       numtextos++;

    closedir(directorio);

    if (numtextos == 0)                          //Si no hay ningún texto, se cierra
la aplicación.
```

```
    {
       printf("No hay archivos disponibles en la carpeta 'Textos'.\nSaliendo de la
aplicación...\n");
       fflush(stdout);
       Decir("no hay archivos disponibles en la carpeta textos. Saliendo de la
aplicación");
       exit(1);
    }
    if (numtextos == 1)                                //Si sólo hay uno, se selecciona
automáticamente
    {
       docu = ObtenerArchivo(0);
       SeleccionarDocumento(docu);
    }
    else
    {
       while(true)                                //Si hay más de uno, se elige el deseado
mediante un menú visual y sonoro
       {
           system("clear");
           cout << "\n    PULSE ESPACIO TRAS ESCUCHAR EL DOCUMENTO DESEADO, O ESCAPE PARA
SALIR";
           cout << "\n
------------------------------------------------------------------------\n\n";
           Decir("pulse espacio tras escuchar el documento deseado, o escape para
salir");
           festival_say_text(" ");
           sleep(1);
           ComprobarTeclado();
           VaciarBufferTeclado();


           for (int conta = 0; conta <= numtextos-1; conta ++)
           {
              docu = ObtenerArchivo(conta);
              nomdocu = strdup(docu.c_str());
              printf("%s\n",nomdocu);
              Decir(docu);
              sleep(1);
              if (ComprobarTeclado() == ' ')
              {
                  SeleccionarDocumento(docu);
                  break;
              }
           }
       }
    }
}

//-------------------------------------------------------------------------------------
//Función para obtener el nombre de un archivo a partir del lugar que ocupa en la
carpeta.
//-------------------------------------------------------------------------------------
string ObtenerArchivo(int numarch)
{
    struct dirent *d;
    const char *direccion = "./Textos";
    int numtextos = -1;

    DIR *directorio = opendir(direccion);
    while ((d=readdir(directorio)) != NULL)
    {
       if (strcmp("..",d->d_name)!=0 && strcmp(".",d->d_name)!=0)
           numtextos++;
       if (numtextos == numarch)
           return d->d_name;
```

```
    }
    closedir(directorio);
}

//-------------------------------------------------------------------------------------
//Función para preparar el documento escogido para su lectura, eligiendo lectura
completa o desde el último punto.
//-------------------------------------------------------------------------------------
void SeleccionarDocumento(string elnombredoc)
{
    char *nombredoc = strdup(elnombredoc.c_str());
    FILE * eldocumento;
    FILE * laguia;
    char direcdoc[512] = "./Textos/";
    char direcguia[512]= "./Guias/";
    char frasechar[512] = "Procesando el documento ";
    int opcion=0, linea=0;
    char charlinea[5];

    strcat(frasechar,nombredoc);
    strcat(direcdoc,nombredoc);
    strcat(direcguia,nombredoc);
    strcat(direcguia,".guia");

    printf("\n%s\n",frasechar);
    fflush(stdout);
    Decir(frasechar);

    laguia = fopen(direcguia,"r");
    if (laguia != NULL)
    {
       printf("Pulse ESPACIO para continuar por donde lo dejó, o INTRO para leer desde el
principio...");
       fflush(stdout);
       Decir("Pulse espacio para continuar por donde lo dejó, o intro para leer desde el
principio");
       VaciarBufferTeclado();
       while(opcion==0)
       {
           sleep(1);
           if (ComprobarTeclado() == ' ')
              opcion = 1;
           if (ComprobarTeclado() == '\n')
              opcion = 2;
           VaciarBufferTeclado();
       }
       if (opcion == 1)
       {
           fgets(charlinea,5,laguia);
           linea = atoi(charlinea);
       }
    }

    eldocumento = fopen(direcdoc,"r");
    LeerDocumento(eldocumento,linea, direcguia);
}

//-------------------------------------------------------------------------------------
//Función que lee el documento seleccionado, desde la línea indicada y previa conversión
a la codificación adecuada.
//-------------------------------------------------------------------------------------
void LeerDocumento(FILE * eldoc, int lalinea, char archiguia[512])
{
    FILE* fichdest;
    FILE* fichref;
    FILE* fichguia;
```

```
    char charslinea[4096];
    bool finfich = false;
    int lineactual = lalinea;

    fichdest = fopen("./Archiaux/leyendo.txt","w");

    for (int contalin = 0; contalin < lineactual; contalin++)
       fgets(charslinea,4096,eldoc);

    do{
       if (fgets(charslinea,4096,eldoc) == NULL)
           finfich = true;
       else
           fputs(charslinea,fichdest);
    } while(!finfich);

    fflush(fichdest);
    fclose(fichdest);

    RefinarTexto();

    system ("iconv --from-code=UTF-8 --to-code=ISO-8859-1 ./Archiaux/leyendoRef.txt
> ./Archiaux/leyendoISO8859.txt -c -s");

    printf("\nIniciando lectura. (Pulse ESCAPE en cualquier momento para finalizar tras
el párrafo en curso y guardar la posición)");
    fflush(stdout);
    Decir("Iniciando lectura. Pulse ESCAPE en cualquier momento para finalizar tras el
párrafo en curso y guardar la posición");
    sleep(1);

    fichref = fopen("./Archiaux/leyendoISO8859.txt","r");
    finfich = false;
    festival_eval_command("(voice_JuntaDeAndalucia_es_pa_diphone)");
    do{
       VaciarBufferTeclado();
       ComprobarTeclado();
       if (fgets(charslinea,4096,fichref) == NULL)
       {
           finfich = true;

           remove(archiguia);
       }
       else
       {
           festival_say_text(charslinea);
           lineactual++;
           fichguia = fopen(archiguia,"w");
           fprintf(fichguia,"%d",lineactual);
           fclose(fichguia);
       }
    }while(!finfich);
    fclose(fichref);
}

//-------------------------------------------------------------------------------------
//Función que comprueba si en las siguientes posiciones al puntero del fichero indicado
se encuentra la cadena.
//-------------------------------------------------------------------------------------
bool ProxCaracteres(FILE * fdesc, string laetiqueta)
{
    char proxcaracter;
    char *etiqueta = strdup(laetiqueta.c_str());
    int posCursorInicial = ftell(fdesc);

    fseek(fdesc,-1,SEEK_CUR);
```

```
    for (int i=0; i <= strlen(etiqueta)-1; i++)
    {
       proxcaracter = fgetc(fdesc);
       if (proxcaracter != etiqueta[i])
       {
           fseek(fdesc,posCursorInicial,SEEK_SET);
           return false;
       }
    }

    return true;
}

//-----------------------------------------------------------------------------------
//Función que comprueba si en las anteriores posiciones al puntero del fichero indicado
se encuentra la cadena.
//-----------------------------------------------------------------------------------
bool AntCaracteres(FILE * fdesc, string laetiqueta)
{
    char proxcaracter;
    char *etiqueta = strdup(laetiqueta.c_str());
    int posCursorInicial = ftell(fdesc);

    fseek(fdesc,-(strlen(etiqueta)+1),SEEK_CUR);
    for (int i=0; i <= strlen(etiqueta)-1; i++)
    {
       proxcaracter = fgetc(fdesc);
       if (proxcaracter != etiqueta[i])
       {
           fseek(fdesc,posCursorInicial,SEEK_SET);
           return false;
       }
    }
    fseek(fdesc,posCursorInicial,SEEK_SET);
    return true;
}

//-----------------------------------------------------------------------------------
//Función que devuelve el número de cifras que aparecen seguidas en las siguientes
posiciones del fichero.
//-----------------------------------------------------------------------------------
int ComprobarSigNumeros(FILE * fdesc)
{
    char proxcaracter;
    int posCursorInicial = ftell(fdesc);
    int numCifras = 0;
    bool haycifras = true;

    do
    {
       proxcaracter = fgetc(fdesc);
       if (proxcaracter >= 48 && proxcaracter <= 57)
       {
           numCifras++;
       }
       else
           haycifras = false;
    } while(haycifras);

    fseek(fdesc,posCursorInicial,SEEK_SET);
    return numCifras;
}

//-----------------------------------------------------
//Función que adecúa el texto para la lectura en Festival.
//-----------------------------------------------------
```

```
void RefinarTexto()
{
    FILE * forigen;
    FILE * fdestino;
    FILE * fdiccionario;
    char caracteractual, ultimoescrito, caracteraux;

    forigen = fopen("./Archiaux/leyendo.txt","r");
    fdestino = fopen("./Archiaux/leyendoRef.txt","w");
    fdiccionario = fopen("./sustituciones","r");

    do{
       caracteractual = fgetc(forigen);
       if (!ProxCaracteres(forigen," ") && !ProxCaracteres(forigen,"«") && !
ProxCaracteres(forigen,"»")) // (Aunque lo primero parece un espacio, es un
multicaracter que no se procesa correctamente y se debe omitir).
       {
           switch(caracteractual)
           {
              case ' ':    // Tratamos aparte los espacios.
                  if (ultimoescrito != ' ' && ultimoescrito != '\n') // Los espacios se
omiten si van precedidos de otro espacio o un salto de linea
                  {
                    if (ProxCaracteres(forigen," .") && ultimoescrito != '.') //o si
después de ellos viene un punto.
                    {
                        fputc('.',fdestino);
                        ultimoescrito = '.';
                    }
                    else if (ProxCaracteres(forigen," (")) // Si después de ellos viene
un paréntesis abierto.
                    {
                        fputc(',',fdestino); // Se pone un punto.
                        ultimoescrito = ',';
                    }
                    else if (ProxCaracteres(forigen," - ")) // Si después de ellos viene
un guión y un espacio.
                    {
                        fputc(',',fdestino); // Se pone un punto.
                        ultimoescrito = ',';
                    }
                    else
                    {
                        fputc(' ',fdestino);  // En otro caso se mantiene el espacio.
                        ultimoescrito = ' ';
                    }
                  }
                  break;
              case '.':    // Tratamos aparte los puntos.
                  if (ultimoescrito >= 48 && ultimoescrito <= 57) // Si el caracter
anterior al punto era un número:
                  {
                    switch(ComprobarSigNumeros(forigen)) //comprobamos si lo que sigue al
punto es una cifra,
                    {
                        case 2: fputc('.',fdestino); //Si la cifra es de 2 dígitos,
ponemos el punto y la siguiente cifra sin espacio, pues
                           fputc(fgetc(forigen),fdestino);  //probablemente es una hora.
                           break;
                        case 3: break;  //Y si es una cifra de 3 dígitos quitamos el
punto porque festival no lo reconoce como separador.
                        default: fputc('.',fdestino);      // En otro caso lo mantenemos.
                           ultimoescrito = '.';
                    }
                  }
                  else if (ultimoescrito != '.' && ultimoescrito != ' ' &&
```

```
ultimoescrito != ',' && ultimoescrito != '\n')
                  {  // Si el caracter anterior al punto no es ni una cifra, ni un punto,
ni un espacio, ni una coma, ni un salto de linea,
                     fputc('.',fdestino);      //lo mantenemos.
                     ultimoescrito = '.';
                  }
                  break;
              case ',':    // Tratamos aparte las comas.
                  if (ultimoescrito >= 48 && ultimoescrito <= 57 &&
ComprobarSigNumeros(forigen) > 0) //si la coma está precedida y seguida de cifras...
                  {  //sustituimos el signo ',' por la palabra 'coma' para que la
pronuncie en lugar de hacer una pausa.
                     fputc(' ',fdestino); fputc('c',fdestino); fputc('o',fdestino);
fputc('m',fdestino); fputc('a',fdestino); fputc(' ',fdestino);
                     ultimoescrito = ' ';
                  }
                  else if (ultimoescrito != ',' && ultimoescrito != ' ')
                  {
                     fputc(',',fdestino);
                     ultimoescrito = ',';
                  }
                  break;
              case '(':    // Los paréntesis los cambiamos por comas para que haga una
pausa
                  {
                     if (ultimoescrito != ',' && ultimoescrito != '.')
                     {
                         fputc(',',fdestino);
                         ultimoescrito = ',';
                     }
                  }
                  break;
              case ')':
                  {
                     if (ProxCaracteres(forigen,").("))
                     {
                         fputc('.',fdestino);
                         ultimoescrito = '.';
                     }
                     else
                     {
                         fputc(',',fdestino);
                         ultimoescrito = ',';
                     }
                  }
                  break;
              default: //Cualquier otro caracter no tratado especialmente lo mostramos
sin más,
                  if (ultimoescrito == '.' || ultimoescrito == ',') //poniendo antes un
espacio si va justo después de un punto o una coma.
                     fputc(' ',fdestino);
                  if (!EncontrarSustitucion(forigen,fdestino,fdiccionario,ultimoescrito))
       //Si no se encuentra una sustitución,
                  {
                     fputc(caracteractual,fdestino);  //escribe el caracter de manera
habitual.
                     ultimoescrito = caracteractual;
                  }
                  else
                     ultimoescrito = '_'; // (cualquier carácter no interceptado en
posteriores iteraciones)
           }
       }
    } while(caracteractual != EOF);

    fflush(fdestino);
```

```
    fclose(forigen);
    fclose(fdestino);
    fclose(fdiccionario);
}

//-------------------------------------------------------------------------------------
//Función que busca si para la próxima palabra hay una pronunciación más adecuada en el
fichero de sustituciones.
//-------------------------------------------------------------------------------------
bool EncontrarSustitucion(FILE * orig, FILE * dest, FILE * dicci, char ultchar)
{
    if (!(ultchar ==' ' || ultchar =='.' || ultchar ==',' || ultchar =='(' || ultchar
==')' || ultchar ==':' || ultchar =='\n'))
       return false;
    char pal_dicci[1440];
    char ultimochar;
    int posEncontrada = 0;
    char palabra[1440];
    int posCursorInicial = ftell(orig);
    bool quedan = true;

    fseek(orig,-1,SEEK_CUR);
    fgets(palabra,1440,orig);

    fseek(orig,posCursorInicial,SEEK_SET);
    fseek(dicci,0,SEEK_SET);

    while(quedan)
    {
       if (fgets(pal_dicci,1440,dicci)) // Vamos comprobando linea a linea del fichero.
       {
           if (tolower(pal_dicci[0]) == tolower(palabra[0])) // Si las palabras empiezan
igual
           {
              for (int i=1; i <= strlen(pal_dicci)-1; i++)
              {
                  if (pal_dicci[i] == '>' && (palabra[i]==' ' || palabra[i]=='.' ||
palabra[i]==',' || palabra[i]==')' || palabra[i]=='\n'))
                  { //si han coincidido todas las letras hasta encontrar '>' significa
que la palabra está contemplada
                     for (int j=i+1; j <= strlen(pal_dicci)-2; j++)
                     {
                         fputc(pal_dicci[j],dest);    //así que se escribe lo que haya en
la parte derecha del '>'
                     }
                     fseek(orig,i-1,SEEK_CUR); //y se actualiza el puntero del fichero de
lectura a la posición justo despúes de la palabra.
                     return true;
                  }
                  else
                  {
                     if (tolower(palabra[i]) != tolower(pal_dicci[i])) //Cuando una letra
no coincide entre las cadenas, se acaba con esa linea
                         break;
                  }
              }
           }
       }
       else
       {
           quedan = false;
       }
    }

    return false;
}
```

```
//------------------------------------------------------------------------
//Función que lee el texto indicado previa conversión al charset ISO-8859-1.
//------------------------------------------------------------------------
void Decir(string eltexto)
{
    char *texto = strdup(eltexto.c_str());
    FILE * archiaux;

    archiaux = fopen("./Archiaux/archiaux","w");
    fputs(texto,archiaux);
    fflush(archiaux);
    fclose(archiaux);
    system ("iconv --from-code=UTF-8 --to-code=ISO-8859-1 ./Archiaux/archiaux >
./Archiaux/archiauxISO8859 -c -s");
    archiaux = fopen("./Archiaux/archiauxISO8859","r");
    remove("./Archiaux/archiaux");
    fgets(texto,1440,archiaux);
    festival_eval_command("(voice_el_diphone)");
    festival_say_text(texto);
    fclose(archiaux);
    remove("./Archiaux/archiauxISO8859");
    festival_wait_for_spooler();
}

//---------------------------------------------------------------------------------
//Función que devuelve la última tecla pulsada o fuerza la salida si ésta era Escape.
//---------------------------------------------------------------------------------
char ComprobarTeclado()
{
    char tecla = UltimaTeclaPulsada();
    if (tecla == 27)
    {
       cout << "\n\nGracias por usar nuestro programa. Vuelva pronto.\n\n";
       Decir("Gracias por usar nuestro programa. Vuelva pronto");
       exit(1);
    }
    return tecla;
}

//-----------------------------------------------
//Función que devuelve la última tecla pulsada.
//-----------------------------------------------
char UltimaTeclaPulsada() // Esta es una versión modificada del Kbhit para linux, para
que devuvelva la última tecla pulsada
{
  struct termios oldt, newt;
  int ch;
  int oldf;

  tcgetattr(STDIN_FILENO, &oldt);
  newt = oldt;
  newt.c_lflag &= ~(ICANON | ECHO);
  tcsetattr(STDIN_FILENO, TCSANOW, &newt);
  oldf = fcntl(STDIN_FILENO, F_GETFL, 0);
  fcntl(STDIN_FILENO, F_SETFL, oldf | O_NONBLOCK);

  ch = getchar();

  tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
  fcntl(STDIN_FILENO, F_SETFL, oldf);

    if(ch != EOF)
    {
       ungetc(ch, stdin);
       return ch;
```

```
    }

  return NULL;
}

//----------------------------------------
//Función para vaciar el buffer de entrada.
//----------------------------------------
void VaciarBufferTeclado()
{
    setvbuf(stdin, NULL, _IONBF,0); // Al deshabilitar la entrada estandar se vacía el
buffer,
    setvbuf(stdin, NULL, _IOFBF,0); //tras lo cual volvemos a habilitarla.
}

//-------------------------------------------------------------------------------------
//Función que comprueba si las voces a utilizar están instaladas y avisa en caso
contrario.
//-------------------------------------------------------------------------------------
void ComprobarVoces()
{
    DIR* directorio;
    bool estavoz1 = false, estavoz2 = false;
    char dirvoz1[72] =
"/usr/share/festival/voices/spanish/JuntaDeAndalucia_es_pa_diphone"; //voz masculina de
la Junta de Andalucía
    char dirvoz2[48] = "/usr/share/festival/voices/spanish/el_diphone"; //voz masculina
estándar
    char carpetavoz1[32] = "JuntaDeAndalucia_es_pa_diphone";
    char carpetavoz2[12] = "el_diphone";

    if ((directorio = opendir(dirvoz1)) != NULL)
      estavoz1 = true;

    if ((directorio = opendir(dirvoz2)) != NULL)
      estavoz2 = true;

    if (!estavoz1 || !estavoz2)
    {
      printf("Algunas voces utilizadas no están instaladas. Esto puede afectar
sensiblemente la calidad de la experiencia:\n");
      if (!estavoz1)
          printf("%s\n",carpetavoz1);
      if (!estavoz2)
          printf("%s\n",carpetavoz2);
      fflush(stdout);
      Decir("algunas voces utilizadas no están instaladas. Esto puede afectar
sensiblemente la calidad de la experiencia");
    }
}

//-------------------------------------------------------------------------------------
//Función que carga el controlador de sonido indicado en el fichero configaudio, que en
caso de no existir lo crea.
//-------------------------------------------------------------------------------------
void InicializarControlador()
{
    FILE * archiconfig;
    char codigodriver;

    archiconfig = fopen("configaudio","r");
    if (archiconfig == NULL)
    {
      codigodriver = BuscaControlador();
      sleep(1);
      if (codigodriver == '0')
```

```
      {
          cout << "\n¡No se encuentra un controlador de sonido adecuado!\nSaliendo de la
aplicación...\n\n";
          exit(1);
      }
      archiconfig = fopen("configaudio","w");
      fputc(codigodriver,archiconfig);
      fclose(archiconfig);
   }
   else
   {
      codigodriver = fgetc(archiconfig);
      fclose(archiconfig);
      switch (codigodriver)
      {
          case '1': festival_eval_command("(Parameter.set 'Audio_Command \"aplay -D
plug:dmix -q -c 1 -t raw -f S16_BE -r $SR $FILE\")"); break;
          case '2': festival_eval_command("(Parameter.set 'Audio_Command \"esdaudio
$FILE\")"); break;
          case '3': festival_eval_command("(Parameter.set 'Audio_Command \"sun16audio
$FILE\")"); break;
          case '4': festival_eval_command("(Parameter.set 'Audio_Command \"sunaudio
$FILE\")"); break;
          case '5': festival_eval_command("(Parameter.set
'Audio_Command \"freebsd16audio $FILE\")"); break;
          case '6': festival_eval_command("(Parameter.set 'Audio_Command \"linux16audio
$FILE\")"); break;
      }
      festival_eval_command("(Parameter.set 'Audio_Method 'Audio_Command)");
      festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
   }
   system("clear");
}

//-------------------------------------------------------------------------------------
//Función que se encarga de ir probando los controladores uno a uno hasta que encuentra
uno válido y lo guarda.
//-------------------------------------------------------------------------------------
char BuscaControlador()
{
   time_t Tantes, Tdespues;

   system("clear");
   cout << "Detectada primera ejecución. Buscando controlador de audio adecuado...";
   fflush(stdout);
   sleep(3);

   festival_eval_command("(Parameter.set 'Audio_Command \"aplay -D plug:dmix -q -c 1 -t
raw -f S16_BE -r $SR $FILE\")");
   festival_eval_command("(Parameter.set 'Audio_Method 'Audio_Command)");
   festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
   time(&Tantes);
   Decir("detectado el controlador ALSA");
   system("clear");
   time(&Tdespues);
   if (difftime(Tdespues,Tantes) >= 2)
      return '1';

   festival_eval_command("(Parameter.set 'Audio_Method 'esdaudio)");
   festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
   time(&Tantes);
   Decir("detectado el controlador E S D.");
   system("clear");
   time(&Tdespues);
   if (difftime(Tdespues,Tantes) >= 2)
      return '2';
```

```
    festival_eval_command("(Parameter.set 'Audio_Method 'sun16audio)");
    festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
    time(&Tantes);
    Decir("detectado el controlador san audio 16 bits");
    system("clear");
    time(&Tdespues);
    if (difftime(Tdespues,Tantes) >= 2)
       return '3';

    festival_eval_command("(Parameter.set 'Audio_Method 'sunaudio)");
    festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
    time(&Tantes);
    Decir("detectado el controlador san audio");
    system("clear");
    time(&Tdespues);
    if (difftime(Tdespues,Tantes) >= 2)
       return '4';

    festival_eval_command("(Parameter.set 'Audio_Method 'freebsd16audio)");
    festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
    time(&Tantes);
    Decir("detectado el controlador frii B S D audio 16 bits");
    system("clear");
    time(&Tdespues);
    if (difftime(Tdespues,Tantes) >= 2)
       return '5';

    festival_eval_command("(Parameter.set 'Audio_Method 'linux16audio)");
    festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
    time(&Tantes);
    Decir("detectado el controlador Linux audio 16 bits");
    system("clear");
    time(&Tdespues);
    if (difftime(Tdespues,Tantes) >= 2)
       return '6';

    return '0';
}
```

5.2_ WEB-A-VOZ

5.2.1_ PROPÓSITO

El objetivo de esta aplicación es ofrecer un lector de noticias web pudiendo elegir de entre una serie de diarios digitales disponibles. El principal logro que esto conlleva es que abre a las personas invidentes o con dificultades visuales severas el acceso a un medio de información al que no tenían posibilidad de acceder, permitiéndoles así estar al tanto de las últimas noticias acontecidas en el momento que lo deseen. Además proporciona una alternativa a aquellos que aunque no tengan ninguna deficiencia visual, prefieren enterarse de las noticias mientras realizan otras actividades que requieran su atención o sencillamente no tengan ganas de leer.

Una vez más, presentaremos las opciones y mensajes del programa de forma tanto auditiva como visual para posibilitar el acceso a personas invidentes y reforzar el entendimiento para las que no lo son.

La selección de los cinco medios digitales disponibles para consultar es casual, siendo el procedimiento extensible a potencialmente cualesquiera otros medios que ofrezcan, al igual que estos, sindicación RSS de sus noticias.

5.2.2_ PRERREQUISITOS

Para asegurar la correcta compilación y ejecución de la aplicación “Web-a-Voz”, mostraremos una lista de las instalaciones previas de software necesarias, así como las versiones utilizadas en nuestro equipo durante las pruebas y la programación de la aplicación. Una vez más, nombraremos todos los paquetes y programas necesarios sin tener en cuenta que se hayan podido instalar anteriormente para probar los ejemplos de secciones anteriores o alguna de las otras aplicaciones que componen este proyecto.

Festival, su API de programación, y voz en castellano
El motor de síntesis del habla Festival, las cabeceras y funciones para su integración en aplicaciones, y la voz oficial en castellano.
Versión utilizada: 1:2.0.95-beta-2ubuntu1
Orden de consola para instalación desde el repositorio:

apt-get install festival festival-dev festvox-ellpc11k

Iconv
El conversor entre juegos de caracteres desde el shell. En nuestra distribución 10.10 de Ubuntu ya se encontraba instalado.
Versión utilizada: 1.7-2
Orden de consola para instalación desde el repositorio:

apt-get install libtext-iconv-perl

Lo ideal habría sido utilizar las librerías para C++ del *iconv* en lugar de depender de la ejecución externa desde consola, mediante la orden *system*. Esto lo propondremos en "Trabajos Futuros" (sección 7).

Wget[61]
La utilidad de red para la obtención de archivos de la web desde el shell. En nuestra distribución 10.10 de Ubuntu ya se encontraba instalada.
Versión utilizada: 1.12-1.1ubuntu3
Orden de consola para instalación desde el repositorio:

apt-get install wget

Voces de la Junta de Andalucía
Las voces de considerable calidad de la Junta de Andalucía, una masculina y otra femenina.
Versión utilizada: 1.0-1
Instalación no disponible desde el repositorio. Ver sección 3.2, página 13.

G++
El compilador GNU de C++.
Versión utilizada: 4:4.4.4-1ubuntu2
Orden de consola para instalación desde el repositorio:

apt-get install g++

Por último recordar que el Sistema Operativo en el que se ha creado y probado la aplicación es:
Ubuntu
Versión: 10.10

En caso de disponer de versiones de software o sistema operativo distintos a los recién indicados, no podemos asegurar que el funcionamiento sea idéntico al descrito en la siguiente subsección.

5.2.3_ FUNCIONAMIENTO

La aplicación completa está disponible en la carpeta "APLICACIONES/WebaVoz" del anexo.

En primer lugar procedamos a compilar la aplicación. De nuevo, basta con situarnos en el directorio correspondiente a ésta desde la consola y ejecutar la herramienta de generación de código introduciendo "make", aprovechando que en nuestras aplicaciones hay incluído un Makefile que contiene la línea necesaria para la compilación:

g++ webavoz.cc -I/usr/lib -l Festival -I/usr/include/festival -I/usr/include/speech_tools/ -leststring -lestbase -lestools -o webavoz

Si todo ha ido bien, ya tendremos el ejecutable "webavoz" listo para ejecutar en el directorio principal. Pero antes vamos a repasar el contenido inicial de la carpeta del programa.

Carpetas:

Archiaux: Contendrá ficheros auxiliares correspondientes a los refinamientos y conversiones de charsets del documento a leer.

Archiweb: Contendrá los ficheros RSS descargados de la webs seleccionadas para su lectura.

Ficheros:

webavoz: Ejecutable de la aplicación.

webavoz.cc: Fichero fuente de la aplicación. En la siguiente sección explicaremos cada una de las funciones que contiene.

Makefile: Fichero con la orden para generar la aplicación a través de la herramienta de generación "make".

sustituciones: fichero que contiene especificaciones de pronunciación alternativa a la literal para una serie de palabras, que el propio usuario puede ampliar con sus propias experiencias. Se detallará detenidamente en la descripción de la función *EncontrarSustitucion* en esta misma sección.

Funcionamiento paso a paso:

En primer lugar ejecutamos el programa escribiendo "./webavoz" en la consola. Si es la primera vez que lo ejecutamos,

un mensaje nos alertará de esto y se procederá a buscar y almacenar el controlador de sonido adecuado. A continuación, se realizará la comprobación de las voces instaladas, que en caso de que falte alguna de las que se utilizarán, avisará de que la experiencia auditiva puede ser de inferior calidad a la esperada. Hasta aquí los pasos seguidos son comunes a las tres aplicaciones desarrolladas, utilizando de hecho las mismas funciones.

En cualquier caso, inmediatamente después se presentará al usuario un menú en el que se van leyendo cíclicamente los nombres de los cinco diarios contemplados en la aplicación. Si se pulsa la barra espaciadora tras escuchar uno de los nombres, se procederá a leer el fichero correspondiente al RSS de dicho diario, convenientemente filtrado y corregido (si un diario disponía de varios RSS, se seleccionó el de 'noticias última hora'). Tras finalizar la lectura de las noticias, el programa vuelve a presentar el menú de selección de opciones, en el que de nuevo se puede volver a seleccionar un diario para escuchar, o bien finalizar la ejecución pulsando Escape.

Descripción de las funciones

Vamos a describir el funcionamiento de cada una de las funciones implicadas en la aplicación, las cuales están disponibles al completo en la siguiente sección (5.2.4). Algunas descripciones son exactamente iguales a las ya expuestas para la aplicación e-Narrador, pero hemos preferido volver a ponerlas para no perder la coherencia y comprensión de las relaciones entre funciones, y no depender de explicaciones de otras aplicaciones para que ésta sea entendible únicamente mirando su propia sección.

(1) → int Main(int argc, char **argv)

Es la función principal, el punto de entrada de nuestra aplicación. Al principio de esta función nos limitamos a hacer como de costumbre:

1- Inicializar Festival.
2- Llamar a la función **InicializarControlador→(2)**.
3- Llamar a la función **ComprobarVoces→(5)**.

Después se muestra el menú principal, escribiendo las opciones y leyéndolas con la función **Decir→(4)**, en el cual va comprobando las posibles teclas pulsadas con **ComprobarTeclado→(6)**, vaciando el buffer entre lectura y lectura de cada opción con **VaciarBufferTeclado→(8)**. Como puede verse, tras leer cada opción se espera un segundo antes de proceder a leer la siguiente para dar tiempo a reaccionar al usuario para que pulse la tecla Espacio. Cuando la comprobación del teclado devuelve 'True' (se ha pulsado la tecla espacio), se borra el contenido de la carpeta

“Archiweb” y se llama a la función correspondiente de obtener y filtrar el diario seleccionado. Estas funciones son: **ObtenerDiarioPublico→(9)**, **ObtenerDiarioLevante→(10)**, **ObtenerDiarioMarca→(11)**, **ObtenerDiarioElPais→(12)**, u **ObtenerDiarioElMundo→(13)**. Una vez comprobado que se ha procesado correctamente la página deseada con **ObtencionExitosa→(14)**, se procede a leerla con **LeerNoticias→(20)**. Cuando finaliza la lectura, se vuelve a mostrar el menú y así continuamente hasta que la función que comprueba el teclado lee la tecla Escape y se finaliza la ejecución.

(2) → void InicializarControlador()

Es la función encargada de cargar el controlador de sonido adecuado para nuestro equipo. Lo que hace es comprobar si existe un fichero llamado “configaudio” en el directorio principal de la aplicación. Si no existe, llama a la función **BuscaControlador→(3)**, que devuelve un código que si es distinto de cero se almacena en el fichero “configaudio” que crea en ese mismo instante, mientras que si es cero se alerta de que no se encuentra un controlador adecuado y se finaliza la aplicación. Después utiliza ese código (entre 1 y 6) para ejecutar una instrucción de Festival que cambia el controlador de sonido a utilizar correspondiente a dicho código. Finalmente limpia la pantalla.

(3) → char BuscaControlador()
-Devuelve: el código del controlador adecuado entre 1 y 6, o un 0 si no encuentra ninguno.

Esta función no hace otra cosa que ir cargando uno a uno los seis controladores de sonido que conocemos, intentando reproducir para cada uno de ellos una frase afirmando que se ha encontrado dicho controlador, utilizando para realizar dicha lectura la función **Decir→(4)**. La aplicación cronometra esos intentos de lectura de cada frase y comprueba el tiempo obtenido; Sólo cuando ese tiempo sea igual o mayor a dos segundos, significará que la frase efectivamente se ha leído, por lo que se considera válido el último controlador cargado y se devuelve su código correspondiente. Si ningún controlador ha sido válido, el valor de retorno será entonces cero.

Hemos tenido que idear una estrategia de detección así de curiosa, porque las funciones de Festival no se han comportado como esperábamos: la función que cambia de controlador devuelve 1 siempre independientemente de si el controlador es válido o no, y la función para leer un texto devuelve 1 tanto si consigue leer como si no.

(4) → void Decir(string eltexto)
-Recibe: La cadena de texto a leer.

El objetivo de esta función es convertir la cadena de texto suministada en un fichero con codificación ISO-8859-1 para evitar los errores de lectura de algunas palabras castellanas con la codificación UTF-8, como comentábamos en la sección 3.3.

En primer lugar se crea un fichero "archiaux" en la carpeta "Archiaux" y se guarda en él la cadena suministrada a la función como argumento. A continuación se hace una conversión de dicho fichero, codificado en UTF-8, a otro llamado "archiauxISO8859" codificado en ISO-8859-1, borrando inmediatamente después el primero. Después se extrae la línea almacenada en el nuevo fichero, se cambia la voz a utilizar a la castellana principal, y se lee dicha línea, tras lo que se borra también el último archivo.

Como se puede ver, las conversiones entre juegos de caracteres se realizan invocando la orden iconv desde consola, utilizando para ellos la instrucción system. Como ya comentábamos en los prerrequisitos, sería preferible utilizar en su lugar las librerias de C++ (ver Trabajos Futuros, sección 7).

(5) → void ComprobarVoces()

En primer lugar, se almacenan en tres arrays las rutas completas de las carpetas donde deberían estar las voces requeridas por la aplicación, y también únicamente el nombre en otros tres. Después sólo hay que intentar abrir los directorios correspondientes a dichas rutas e indicar si el intento ha sido fallido con alguno, lo que significará que no ha sido instalado, y avisará de la sensible pérdida de calidad de la experiencia debido a ello.

(6) → char ComprobarTeclado()
-Devuelve: la última tecla pulsada cuando ésta sea distinta de Escape.

Esta función agrupa la obtención de la última tecla pulsada desde la última limpieza del buffer, con **UltimaTeclaPulsada→(7)**, y la salida del programa cuando esta tecla sea Escape. Esta agrupación evita un exceso de comprobaciones en los menús.

(7) → char UltimaTeclaPulsada()
-Devuelve: la última tecla pulsada.

Esta función es una versión modificada de una versión de la función C++ de Borland *Kbhit*, propuesta por el usuario *itsme86* en

el foro *cprogramming*. La función en concreto se encuentra en el siguiente enlace:
http://cboard.cprogramming.com/c-programming/63166-kbhit-linux.html

La modificación realizada al final de la función permite que en lugar de saber únicamente si se ha pulsado una tecla cualquiera o no, sepamos la tecla exacta.

(8) → void VaciarBufferTeclado()

Esta escueta función únicamente realiza dos acciones: la línea superior sirve para deshabilitar la entrada estándar, lo cual a su vez vacía el buffer correspondiente. La segunda línea vuelve a habilitarla. La función es necesaria porque la comprobación de teclado realizada por las funciones del teclado no vacía el buffer.

Las siguientes cinco funciones, relativas a la obtención y filtrado de los ficheros de noticias, las describiremos en común por ser estructuralmente idénticas, difiriendo sólo en algunas etiquetas a reconocer y filtrar. Estas funciones son:
(9)→ void ObtenerDiarioPublico()
(10)→ void ObtenerDiarioLevante()
(11)→ void ObtenerDiarioMarca()
(12)→ void ObtenerDiarioElPais()
(13)→ void ObtenerDiarioElMundo()

En estas funciones se realizan dos acciones. En primer lugar, utilizando la utilidad externa *wget* desde consola se descarga de internet el RSS de la web del diario correspondiente, que no es más que un fichero con estructura de etiquetas XML, pensado para su acceso desde un navegador convencional, pero que nosotros desglosaremos para nuestros fines. Estos archivos, en orden según la lista de funciones que hemos enumerado, son "index.html", "levante.xml", "portada.xml", "feedId=1022", o "portada.xml". No nos debe abrumar la variedad de extensiones (xml, html o ninguna) que presentan los ficheros, pues en realidad si los examinamos vemos que todos tienen prácticamente la misma estructura XML.

A continuación, se procede a extraer del fichero descargado la información referente a las noticias. Para ello se crea un fichero "textovoz.txt" dentro del directorio "Archiweb" al que se irá copiando únicamente el texto referente a los titulares y las noticias, ignorando otros datos, etiquetas y símbolos no legibles. Para indicar cuándo se debe copiar el texto de la posición actual y cuándo no, se usa una variable booleana *escribiendo* a modo de flag que será 'True' cuando debe copiarse el texto, y 'False' cuando no. Para ello, con la función **ProxCaracteres→(15)** se identifican las etiquetas XML que se utilizan para indicar que

empieza texto legible de titulares (p.e. <item><title>) o cuerpos de noticia (p.e. <description>), lo cual activa el flag *escribiendo* para que comience a copiar el texto que vaya encontrando hasta que se encuentren etiquetas de cierre (p.e. </title> o </description>) momento en el que se desactiva el flag. También es requisito indispensable para escribir que haya más etiquetas de apertura que de cierre, para prevenir problemas con posibles anidamientos. Además de esto suele prevenirse la aparición de caracteres o etiquetas no deseados (p.e. <p> o &amp).

Cuando se termina de recorrer el fichero de noticias, es decir, se obtiene una línea que contenga el EOF (end of file), se cierran ese fichero y el "textovoz.txt" y se llama a la función **RefinarTexto→(16)** pasándole el número de líneas iniciales que habrá que ignorar según el diario del que se hayan extraído los datos.

(14) → bool ObtencionExitosa()
-Devuelve: 'True' cuando el fichero "textovozRef.txt" existe, 'False' en caso contrario.

Sencilla función que únicamente comprueba si existe el fichero "textovozRef.txt" dentro de la carpeta "Archiaux", lo cual significaría que el fichero de noticias esperado se obtuvo de la web y se procesó de la manera esperada, devolviendo 'True' si es así, o 'False' en caso contrario.

(15) → bool ProxCaracteres(FILE * fdesc, string laetiqueta)
-Recibe: el descriptor del fichero de lectura y el texto a comprobar.
-Devuelve: 'True' cuando la cadena recibida se encuentra a partir de la posición actual del cabezal de lectura en el fichero, o 'False' en caso contrario.

Incialmente se guarda la posición del puntero de lectura del fichero, y se procede a comparar los siguientes caracteres uno a uno con los del string suministrado. Si alguno de ellos no coincide, se devuelve 'False' y se devuelve el puntero de lectura a su posición inicial, mientras que si todas las comparaciones tienen éxito se devuelve 'True'.

(16) → void RefinarTexto(int lineasinutiles)
-Recibe: Número de líneas iniciales a ignorar.

Esta función tiene como objetivo corregir y mejorar el archivo de texto creado a partir de la noticia seleccionada en el menú principal. Debido a su complejidad, el propio código contiene comentarios explicativos de las comprobaciones y sustituciones que

se realizan en cada línea.

En primer lugar, se ignoran tantas líneas iniciales como indique el argumento recibido. A continuación, básicamente, el puntero de lectura del fichero origen “textovoz.txt” va copiando al fichero destino “textovozRef.txt” los caracteres que encuentra secuencialmente, excepto en las comprobaciones que va realizando, como al encontrar caracteres extraños, paréntesis, espacios o signos de puntuación excesivos, etc. En definitiva cualquier carácter que tras realizar pruebas establecimos que Festival ignora o no pronuncia adecuadamente. Algunas comprobaciones necesitan consultar los caracteres que aparecerán más adelante con **ProxCaracteres→(15)**, caracteres anteriores con **AntCaracteres→(17),** u obtener cuántas cifras consecutivas aparecen con **ComprobarSigNumeros→(18)** para establecer si por ejemplo se debe mantener el punto separador de miles. Además, se comprobará si en el texto hay presentes palabras que contemplamos en el fichero “sustituciones”, con **EncontrarSustitucion→(19)**.

(17) → bool AntCaracteres(FILE * fdesc, string laetiqueta)
-Recibe: el descriptor del fichero de lectura y el texto a comprobar.
-Devuelve: 'True' cuando la cadena recibida se encuentra a partir de la posición actual del cabezal de lectura del fichero, o 'False' en caso contrario.

De forma análoga a la función anterior, ésta almacena la posición actual del puntero de lectura y procede a comparar carácter a carácter, pero esta vez hacia atrás. Si alguna comparación falla devuelve 'False' y si todas tienen éxito devuelve 'True'. En cualquiera de los dos casos devuelve el puntero de lectura a la posición almacenada inicialmente.

(18) → int ComprobarSigNumeros(FILE * fdesc)
-Recibe: el descriptor del fichero de lectura.
-Devuelve: el número de caracteres numéricos existentes desde la posición actual.

Esta función guarda la posición actual del puntero de lectura del fichero, y procede a analizar las siguientes posiciones hasta que encuentre un carácter que no sea una cifra, manteniendo el recuento de éstas. Finalmente se devuelve el puntero de lectura a la posición guardada inicialmente y se devuelve el número de cifras consecutivas.

(19) → bool EncontrarSustitucion(FILE * orig, FILE * dest, FILE * dicci, char ultchar)
-Recibe: el descriptor del fichero de origen, el descriptor del

fichero destino, el descriptor del fichero de sustituciones y el último carácter encontrado en el fichero de origen.
-Devuelve: Si se ha aplicado alguna sustitución ('True) o no ('False').

Esta función fue creada explícitamente para esta aplicación, pero rápidamente pudimos comprobar que es un mecanismo de corrección potente para cualquier aplicación TTS que se nos ocurra. Nos permite utilizar un fichero llamado "sustituciones", ampliable por cada usuario según sus experiencias, que sustituye palabras que leídas literalmente no suenan como deberían por una escritura lo más aproximada a cómo deberían pronunciarse, como es el caso de abreviaturas, siglas o nombres extranjeros. Por ejemplo, si el lector encontrara la palabra "Michael" y la leyera como de costumbre, la pronunciación sería tal cual "mi-cha-el" como si de una palabra española más se tratara, pero con una entrada de sustitución adecuada en el fichero de la siguiente forma: "michael>máiquel" le indicamos que cuando encuentre esa palabra no debe leerla así, sino como se especifica a la derecha del carácter ">". En la carpeta "EJEMPLOS/SUSTITUCIONES" del anexo hay disponibles algunas noticias ficticias creadas explícitamente para contener varias de las palabras contempladas en el fichero de sustituciones, junto con dos archivos de audio, resultado de leerlas utilizando dicho fichero ("CON_sustituciones.wav") y sin utilizarlo ("SIN_sustituciones.wav"). Como se puede apreciar, la diferencia es más que notable. Veamos cómo funciona:

En primer lugar, comprueba si el último carácter leído no era una letra, con lo cual ya se sabe que no se aplicará una sustitución y se devuelve 'False'. En otro caso, se almacenan los siguientes caracteres del fichero origen, y se van comparando con las entradas del fichero de sustituciones línea a línea. Para cada una de las líneas se hace una comparación carácter a carácter, hasta que una comparación resulta fallida, lo cual hace que se pase a la siguiente línea, o hasta encontrar el carácter ">" en el fichero de sustituciones y un espacio, punto, coma, paréntesis o cambio de línea en la misma posición en el fichero de origen, lo cual significa que la palabra comprobada tiene una sustitución contemplada. En este último caso se procede a escribir en el fichero destino lo que haya a continuación del ">" en el fichero de sustituciones, se sitúa el puntero de posición justo después de la palabra sustituída y se devuelve 'True'. Si ninguna sustitución ha sido posible después de comprobar todas las líneas, se devuelve 'False'.

(20) → void LeerNoticias()

Aquí es donde finalmente se procederá a leer las noticias tras todos los filtrados, refinamientos y sustituciones que deberían haber dado como resultado un texto bastante aceptable

para su lectura. Antes, se necesita transformar el fichero de texto obtenido de codificación UTF-8 a la codificación ISO-8859-1, que como comentábamos en los primeros pasos de Festival en la sección 3.3, es la codificación adecuada para representar algunos caracteres de nuestro idioma como las tildes o la 'ñ' que el Unicode no contempla. Así pues, pasamos de tener el fichero "textovozRef.txt" en UTF-8 a tener el fichero "textovozISO8859.txt" en ISO-8859-1, ambos en la carpeta "Archiaux". Este último fichero ya será el que se procederá a leer.

Seguidamente se llega al bucle de lectura, que repite cíclicamente las siguientes acciones: establece como voz a utilizar la masculina de la Junta de Andalucía, obtiene una linea del archivo "textovozISO8859.txt", la lee, vuelve a obtener una línea, y también la lee. Entonces cambia la voz a la femenina de la Junta de Andalucía y vuelve a repetir el proceso de obtener 2 líneas y leerlas de manera idéntica, tras lo cual se vuelve a repetir el proceso volviendo a cargar la voz masculina y el resto de acciones mientras queden líneas para leer en el fichero. Con esto lo que se consigue es que la voz masculina y femenina vayan leyendo alternadamente una noticia cada uno, aprovechando que cada noticia consta de dos líneas: una para el título, y otra para el cuerpo. El resultado final es una lectura menos monótona y en la que es más fácil distinguir entre el fin de una noticia y el comienzo de la siguiente.

5.2.4_ CÓDIGO FUENTE

A continuación se expone el código fuente de la aplicación "e-Narrador" al completo:

```
#include <stdio.h>
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <festival.h>
#include <sstream>
#include <string>
#include <termios.h>
#include <fcntl.h>
#include <dirent.h>

void ObtenerDiarioPublico(), ObtenerDiarioMarca(), ObtenerDiarioLevante(),
ObtenerDiarioElPais(), ObtenerDiarioElMundo();
void InicializarControlador(), ComprobarVoces(), RefinarTexto(int), LeerNoticias(),
Decir(string), VaciarBufferTeclado();
bool ProxCaracteres(FILE*,string), AntCaracteres(FILE*,string), ObtencionExitosa(),
EncontrarSustitucion(FILE*,FILE*,FILE*,char), ComprobarTeclado();
int ComprobarSigNumeros(FILE*);
char BuscaControlador(), UltimaTeclaPulsada();

int heap_size = 210000;
int load_init_files = 1;
```

```
int main(int argc, char **argv)
{
    bool salir = false, haleido = false;
    char opcionchar;
    int opcionint;
    festival_initialize(load_init_files,heap_size);

    InicializarControlador();

    ComprobarVoces();

    do {
       while(!salir){
           system("clear");
           cout << "\n    PULSE ESPACIO TRAS ESCUCHAR EL DIARIO DESEADO, O ESCAPE PARA
SALIR";
            cout << "\n
----------------------------------------------------------------------\n\n";
           Decir("pulse espacio tras escuchar el diario deseado, o escape para salir");
           sleep(1);
           ComprobarTeclado();

           VaciarBufferTeclado();
           cout << "> Diario Público (www.publico.es) [5 seg. aprox]\n\n";
           Decir("diario público");
           sleep(1);
           if (ComprobarTeclado())
           {
              opcionint = 1;
              break;
           }

           VaciarBufferTeclado();
           cout << "> Diario Levante (www.levante-emv.com) [20 seg. aprox]\n\n";
           Decir("Diario Levante");
           sleep(1);
           if (ComprobarTeclado())
           {
              opcionint = 2;
              break;
           }

           VaciarBufferTeclado();
           cout << "> Diario Marca (www.marca.com) [5 seg. aprox]\n\n";
           Decir("Diario Marca");
           sleep(1);
           if (ComprobarTeclado())
           {
              opcionint = 3;
              break;
           }

           VaciarBufferTeclado();
           cout << "> Diario El País (www.elpais.com) [5 seg. aprox]\n\n";
           Decir("Diario el país");
           sleep(1);
           if (ComprobarTeclado())
           {
              opcionint = 4;
              break;
           }

           VaciarBufferTeclado();
           cout << "> Diario El Mundo (www.elmundo.es) [5 seg. aprox]\n\n";
           Decir("Diario el mundo");
```

```
            sleep(1);
            if (ComprobarTeclado())
            {
               opcionint = 5;
               break;
            }
        }

        cout << "\nProcesando. Por favor espere...\n\n";
        Decir("procesando. Por favor espere");

        system("rm -f ./Archiweb/*");    //vaciamos el contenido de los archivos derivados
de las webs

        switch (opcionint){
            case 1: ObtenerDiarioPublico(); break;
            case 2: ObtenerDiarioLevante(); break;
            case 3: ObtenerDiarioMarca(); break;
            case 4: ObtenerDiarioElPais(); break;
            case 5: ObtenerDiarioElMundo(); break;
        }

        if (ObtencionExitosa())
        {
            system("clear");
            LeerNoticias();
            haleido = true;
        }
        else
        {
            cout << "\n\nSe ha producido un error de acceso. Intente este diario más
tarde.\n\n";
            Decir("Se ha producido un error de acceso. Intente este diario más tarde");
        }
    } while(true);
}

//-------------------------------------------------------------------------------
//Función que verifica si se ha obtenido y procesado el fichero deseado desde la web
//-------------------------------------------------------------------------------
bool ObtencionExitosa()
{
    FILE * fichero;
    fichero = fopen("Archiaux/textovozRef.txt","r");
    if (fichero != NULL)
    {
       fclose(fichero);
       return true;
    }
    else
       return false;
}

//-------------------------------------------------------------------------------
//Función que obtiene el RSS del diario Público y extrae de él el texto legiblbe
//-------------------------------------------------------------------------------
void ObtenerDiarioPublico()
{
    FILE * forigen;
    FILE * fdestino;
    char caracter;
    bool escribiendo = false;
    int aperturas = 0; // &lt;
    int cierres = 0; // &gt;

    system("wget -i -D http://www.publico.es/rss/ -o ./Archiaux/log.txt --directory-
```

```
prefix=Archiweb"); //se obtiene index.html

    forigen = fopen("Archiweb/index.html","r");

    if (forigen != NULL)
    {
       fdestino = fopen("Archiaux/textovoz.txt","w");
       do{
           caracter = fgetc(forigen);
           if (!escribiendo)
           {
              if (ProxCaracteres(forigen,"<item><title>") ||
ProxCaracteres(forigen,"<description>"))
                  escribiendo = true;
           }
           else
           {
              if (ProxCaracteres(forigen,"</title>") ||
ProxCaracteres(forigen,"</description>"))
              {
                  escribiendo = false;
                  fputc('.',fdestino);
                  fputc('\n',fdestino);
              }
              else if (ProxCaracteres(forigen,"&lt;p&gt;"))
              {
                  if (aperturas == cierres)
                  {
                     fputc('.',fdestino);
                     fputc(' ',fdestino);
                  }
              }
              else if (ProxCaracteres(forigen,"&lt;"))
              {
                  aperturas++;
              }

              else if (ProxCaracteres(forigen,"&gt;"))
              {
                  cierres++;
              }
              else if (aperturas <= cierres)
              {
                  if (ProxCaracteres(forigen,"<p>"))
                     fputc('.',fdestino);
                  else if (ProxCaracteres(forigen,"&amp;"))
                     fputc(' ',fdestino);
                  else if (caracter == '-' || caracter =='?')
                     fputc(' ',fdestino);
                  else if (caracter != '"' && caracter != 39 && caracter != '[' &&
caracter != ']')
                     fputc(caracter,fdestino);
              }
           }
       } while(caracter != EOF); //hasta encontrar el fin de linea

       fflush(fdestino);
       fclose(forigen);
       fclose(fdestino);
       RefinarTexto(1); //refinar el texto ignorando la primeras linea
    }
}

//------------------------------------------------------------------------
//Función que obtiene el RSS del diario Levante y extrae de él el texto legiblbe
//------------------------------------------------------------------------
```

```
void ObtenerDiarioLevante()
{
    FILE * forigen;
    FILE * fdestino;
    char caracter;
    bool escribiendo = false;
    system("wget -i -D http://www.levante-emv.com/elementosInt/rss/AlMinuto -o
./Archiaux/log.txt --directory-prefix=Archiweb"); //se obtiene AlMinuto
    system ("iconv --from-code=ISO-8859-1 --to-code=UTF-8 ./Archiweb/AlMinuto >
./Archiaux/levante.xml -c -s"); //lo convertimos a UTF-8 para que el proceso de
refinamiento lo pueda tratar como a cualquier otro usando la tabla ascii estandar
    forigen = fopen("Archiaux/levante.xml","r");

    if (forigen != NULL)
    {
       fdestino = fopen("Archiaux/textovoz.txt","w");
       do{
           caracter = fgetc(forigen);
           if (!escribiendo)
           {
              if (ProxCaracteres(forigen,"<title><![CDATA[") ||
ProxCaracteres(forigen,"<description><![CDATA["))
                  escribiendo = true;
           }
           else
           {
              if (ProxCaracteres(forigen,"]]></title>") ||
ProxCaracteres(forigen,"]]></description>"))
              {
                  escribiendo = false;
                  fputc('.',fdestino);
                  fputc('\n',fdestino);
              }
              else
              {
                  if (caracter == '-' || caracter =='?')
                     fputc(' ',fdestino);
                  else
                     if (caracter != '"' && caracter != 39 && caracter != '[' &&
caracter != ']')
                          fputc(caracter,fdestino);
              }
           }
       } while(caracter != EOF); //hasta encontrar el fin de linea

       fflush(fdestino);
       fclose(forigen);
       fclose(fdestino);
       RefinarTexto(4); //refinar el texto ignorando la primeras linea
    }
}

//--------------------------------------------------------------------------
//Función que obtiene el RSS del diario MARCA y extrae de él el texto legiblbe
//--------------------------------------------------------------------------
void ObtenerDiarioMarca()
{
    FILE * forigen;
    FILE * fdestino;
    char caracter;
    bool escribiendo = false;
    int aperturas = 0; // &lt;
    int cierres = 0; // &gt;

    system("wget -i -D http://estaticos.marca.com/rss/portada.xml -o ./Archiaux/log.txt
--directory-prefix=Archiweb"); //se obtiene portada.xml
```

```
    forigen = fopen("Archiweb/portada.xml","r");

    if (forigen != NULL)
    {
       fdestino = fopen("Archiaux/textovoz.txt","w");
       do{
           caracter = fgetc(forigen);
           if (!escribiendo)
           {
              if (ProxCaracteres(forigen,"<item><title>") ||
ProxCaracteres(forigen,"<description>"))
                  escribiendo = true;
           }

           else
           {
              if (ProxCaracteres(forigen,"</title>") ||
ProxCaracteres(forigen,"</description>"))
              {
                  escribiendo = false;
                  fputc('.',fdestino);
                  fputc('\n',fdestino);
              }

              else if (ProxCaracteres(forigen,"&amp;"))
              {
                  fputc('.',fdestino);
                  fputc('\n',fdestino);


                  escribiendo = false;
              }

              else if (ProxCaracteres(forigen,"&lt;"))
              {
                  aperturas++;
              }

              else if (ProxCaracteres(forigen,"&gt;"))
              {
                  cierres++;
              }
              else if (aperturas <= cierres)
              {
                  if (caracter == '-' || caracter =='?')
                     fputc(' ',fdestino);
                  else
                     if (caracter != '"' && caracter != 39 && caracter != '[' &&
caracter != ']')
                          fputc(caracter,fdestino);
              }
           }
       } while(caracter != EOF); //hasta encontrar el fin de linea


       fflush(fdestino);
       fclose(forigen);
       fclose(fdestino);
       RefinarTexto(2); //refinar el texto ignorando la primeras linea
    }
}

//---------------------------------------------------------------------------
//Función que obtiene el RSS del diario El País y extrae de él el texto legiblbe
//---------------------------------------------------------------------------
```

```
void ObtenerDiarioElPais()
{
    FILE * forigen;
    FILE * fdestino;
    char caracter;
    bool escribiendo = false;
    int aperturas = 0; // &lt;
    int cierres = 0; // &gt;
    system("wget -i -D http://www.elpais.com/rss/feed.html?feedId=1022 -o
./Archiaux/log.txt --directory-prefix=Archiweb"); //feedId=1022
    system ("iconv --from-code=ISO-8859-1 --to-code=UTF-8 ./Archiweb/feed.html?
feedId=1022 > ./Archiaux/elpais.xml -c -s"); //lo convertimos a UTF-8 para que el
proceso de refinamiento lo pueda tratar como a cualquier otro usando la tabla ascii
estandar
    forigen = fopen("Archiaux/elpais.xml","r");

    if (forigen != NULL)
    {
       fdestino = fopen("Archiaux/textovoz.txt","w");
       do{
           caracter = fgetc(forigen);
           if (!escribiendo)
           {
              if (ProxCaracteres(forigen,"<title><![CDATA[") ||
ProxCaracteres(forigen,"<description><![CDATA["))
                  escribiendo = true;
           }
           else
           {
              if (ProxCaracteres(forigen,"]]></title>") ||
ProxCaracteres(forigen,"]]></description>"))
              {
                  escribiendo = false;
                  fputc('.',fdestino);
                  fputc('\n',fdestino);
              }
              else if (ProxCaracteres(forigen,"<"))
              {
                  aperturas++;
              }

              else if (ProxCaracteres(forigen,">"))
              {
                  cierres++;
              }
              else if (aperturas <= cierres)
              {
                  if (caracter == '-' || caracter =='?')
                     fputc(' ',fdestino);
                  else
                     if (caracter != '"' && caracter != 39 && caracter != '[' &&
caracter != ']')
                         fputc(caracter,fdestino);
              }
           }
       } while(caracter != EOF); //hasta encontrar el fin de linea

       fflush(fdestino);
       fclose(forigen);
       fclose(fdestino);
       RefinarTexto(4); //refinar el texto ignorando la primeras linea
    }
}

//------------------------------------------------------------------------- --
//Función que obtiene el RSS del diario El Mundo y extrae de él el texto legiblbe .
```

```
//---------------------------------------------------------------------------- --
void ObtenerDiarioElMundo()
{
    FILE * forigen;
    FILE * fdestino;
    char caracter;
    bool escribiendo = false;
    int aperturas = 0; // &lt;
    int cierres = 0; // &gt;
    system("wget -i -D http://elmundo.feedsportal.com/elmundo/rss/portada.xml -o
./Archiaux/log.txt --directory-prefix=Archiweb"); //se obtiene portada.xml
    forigen = fopen("Archiweb/portada.xml","r");

    if (forigen != NULL)
    {
       fdestino = fopen("Archiaux/textovoz.txt","w");
       do{
           caracter = fgetc(forigen);
           if (!escribiendo)
           {
              if (ProxCaracteres(forigen,"<item><title>") ||
ProxCaracteres(forigen,"<description>"))
                  escribiendo = true;
           }
           else
           {
              if (ProxCaracteres(forigen,"</title>") ||
ProxCaracteres(forigen,"</description>"))
              {
                  escribiendo = false;
                  fputc('.',fdestino);
                  fputc('\n',fdestino);
              }
              else if (ProxCaracteres(forigen,"#160"))
              {
                  if (aperturas == cierres)
                  {
                     fputc('\n',fdestino);
                     escribiendo = false;
                  }
              }
              else if (ProxCaracteres(forigen,"#34"))
              {
                  if (aperturas == cierres)
                  {
                     fputc(' ',fdestino);
                  }
              }
              else if (ProxCaracteres(forigen,"&lt;"))
              {
                  aperturas++;
              }

              else if (ProxCaracteres(forigen,"&gt;"))
              {
                  cierres++;
              }
              else if (aperturas <= cierres)
              {
                  if (ProxCaracteres(forigen,"<p>"))
                     fputc('.',fdestino);
                  else if (ProxCaracteres(forigen,"&amp;"))
                     fputc(' ',fdestino);
                  else if (caracter == '-' || caracter =='?')
                     fputc(' ',fdestino);
                  else if (caracter != '"' && caracter != 39 && caracter != '[' &&
```

```
caracter != ']')
                    fputc(caracter,fdestino);
                }
            }
       } while(caracter != EOF); //hasta encontrar el fin de linea

       fflush(fdestino);
       fclose(forigen);
       fclose(fdestino);
       RefinarTexto(2); //refinar el texto ignorando la primeras linea
    }
}

//-------------------------------------------------------------------------------------
//Función que comprueba si en las siguientes posiciones al puntero del fichero indicado
se encuentra la cadena.
//-------------------------------------------------------------------------------------
bool ProxCaracteres(FILE * fdesc, string laetiqueta)
{
    char proxcaracter;
    char *etiqueta = strdup(laetiqueta.c_str());
    int posCursorInicial = ftell(fdesc);

    fseek(fdesc,-1,SEEK_CUR);
    for (int i=0; i <= strlen(etiqueta)-1; i++)
    {
       proxcaracter = fgetc(fdesc);
       if (proxcaracter != etiqueta[i])
       {
           fseek(fdesc,posCursorInicial,SEEK_SET);
           return false;
       }
    }

    return true;
}

//-------------------------------------------------------------------------------------
//Función que comprueba si en las anteriores posiciones al puntero del fichero indicado
se encuentra la cadena.
//-------------------------------------------------------------------------------------
bool AntCaracteres(FILE * fdesc, string laetiqueta)
{
    char proxcaracter;
    char *etiqueta = strdup(laetiqueta.c_str());
    int posCursorInicial = ftell(fdesc);

    fseek(fdesc,-(strlen(etiqueta)+1),SEEK_CUR);
    for (int i=0; i <= strlen(etiqueta)-1; i++)
    {
       proxcaracter = fgetc(fdesc);
       if (proxcaracter != etiqueta[i])
       {
           fseek(fdesc,posCursorInicial,SEEK_SET);
           return false;
       }
    }
    fseek(fdesc,posCursorInicial,SEEK_SET);
    return true;
}

//-------------------------------------------------------------------------------------
//Función que devuelve el número de cifras que aparecen seguidas en las siguientes
posiciones del fichero.
//-------------------------------------------------------------------------------------
int ComprobarSigNumeros(FILE * fdesc)
```

```
{
    char proxcaracter;
    int posCursorInicial = ftell(fdesc);
    int numCifras = 0;
    bool haycifras = true;

    do
    {
       proxcaracter = fgetc(fdesc);
       if (proxcaracter >= 48 && proxcaracter <= 57)
       {
           numCifras++;
       }
       else
           haycifras = false;
    } while(haycifras);

    fseek(fdesc,posCursorInicial,SEEK_SET);
    return numCifras;
}

//-------------------------------------------------------
//Función que adecúa el texto para la lectura en Festival.
//-------------------------------------------------------
void RefinarTexto(int lineasinutiles)
{
    FILE * forigen;
    FILE * fdestino;
    FILE * fdiccionario;

    char caracteractual, ultimoescrito, caracteraux;

    forigen = fopen("Archiaux/textovoz.txt","r");
    fdestino = fopen("Archiaux/textovozRef.txt","w");
    fdiccionario = fopen("sustituciones","r");

    int numlineas = 0;
    while(numlineas < lineasinutiles)
    {
       caracteractual = fgetc(forigen);
       if (caracteractual == '\n')
           numlineas++;
    }

    do{
       caracteractual = fgetc(forigen);
       if (!ProxCaracteres(forigen," ") && !ProxCaracteres(forigen,"«") && !
ProxCaracteres(forigen,"»")) // (Aunque lo primero parece un espacio, es un
multicaracter que no se procesa correctamente y se debe omitir).
       {
           switch(caracteractual)
           {
              case ' ':    // Tratamos aparte los espacios.
                  if (ultimoescrito != ' ' && ultimoescrito != '\n') // Los espacios se
omiten si van precedidos de otro espacio o un salto de linea
                  {
                     if (ProxCaracteres(forigen," .") && ultimoescrito != '.') //o si
después de ellos viene un punto.
                     {
                         fputc('.',fdestino);
                         ultimoescrito = '.';
                     }
                     else if (ProxCaracteres(forigen," (")) // Si después de ellos viene
un paréntesis abierto.
                     {
                         fputc(',',fdestino); // Se pone un punto.
```

```
                    ultimoescrito = ',';
                }
                else if (ProxCaracteres(forigen," - ")) // Si después de ellos viene
un guión y un espacio.
                {
                    fputc(',',fdestino); // Se pone un punto.
                    ultimoescrito = ',';
                }
                else
                {
                    fputc(' ',fdestino);  // En otro caso se mantiene el espacio.
                    ultimoescrito = ' ';
                }
            }
            break;
        case '.':    // Tratamos aparte los puntos.
            if (ultimoescrito >= 48 && ultimoescrito <= 57) // Si el caracter
anterior al punto era un número:
            {
              switch(ComprobarSigNumeros(forigen)) //comprobamos si lo que sigue al
punto es una cifra,
                {
                    case 2: fputc('.',fdestino); //Si la cifra es de 2 dígitos,
ponemos el punto y la siguiente cifra sin espacio, pues
                       fputc(fgetc(forigen),fdestino);  //probablemente es una hora.
                       break;
                    case 3: break;  //Y si es una cifra de 3 dígitos quitamos el
punto porque festival no lo reconoce como separador.
                    default: fputc('.',fdestino);      // En otro caso lo mantenemos.
                       ultimoescrito = '.';
                }
            }
            else if (ultimoescrito != '.' && ultimoescrito != ' ' &&
ultimoescrito != ',' && ultimoescrito != '\n')
            {  // Si el caracter anterior al punto no es ni una cifra, ni un punto,
ni un espacio, ni una coma, ni un salto de linea,
              fputc('.',fdestino);      //lo mantenemos.
              ultimoescrito = '.';
            }
            break;
        case ',':    // Tratamos aparte las comas.
            if (ultimoescrito >= 48 && ultimoescrito <= 57 &&
ComprobarSigNumeros(forigen) > 0) //si la coma está precedida y seguida de cifras...
            {  //sustituimos el signo ',' por la palabra 'coma' para que la
pronuncie en lugar de hacer una pausa.
              fputc(' ',fdestino); fputc('c',fdestino); fputc('o',fdestino);
fputc('m',fdestino); fputc('a',fdestino); fputc(' ',fdestino);
              ultimoescrito = ' ';
            }
            else if (ultimoescrito != ',' && ultimoescrito != ' ')
            {
              fputc(',',fdestino);
              ultimoescrito = ',';
            }
            break;
        case '(':    // Los paréntesis los cambiamos por comas para que haga una
pausa
            {
              if (ultimoescrito != ',' && ultimoescrito != '.')
              {
                  fputc(',',fdestino);
                  ultimoescrito = ',';
              }
            }
            break;
        case ')':
```

```
                {
                   if (ProxCaracteres(forigen,"."))
                   {
                       fputc('.',fdestino);
                       ultimoescrito = '.';
                   }
                   else
                   {
                       fputc(',',fdestino);
                       ultimoescrito = ',';
                   }
                }
                break;
             default: //Cualquier otro caracter no tratado especialmente lo mostramos
sin más,
                if (ultimoescrito == '.' || ultimoescrito == ',') //poniendo antes un
espacio si va justo después de un punto o una coma.
                   fputc(' ',fdestino);
                if (!EncontrarSustitucion(forigen,fdestino,fdiccionario,ultimoescrito))
                {
                   fputc(caracteractual,fdestino);
                   ultimoescrito = caracteractual;
                }
                else
                   ultimoescrito = '_'; // (cualquier carácter no interceptado en
posteriores iteraciones)
          }
       }
    } while(caracteractual != EOF);

    fflush(fdestino);
    fclose(forigen);
    fclose(fdestino);
    fclose(fdiccionario);
}

//-----------------------------------------------------------------------------------
//Función que busca si para la próxima palabra hay una pronunciación más adecuada en el
fichero de sustituciones.
//-----------------------------------------------------------------------------------
bool EncontrarSustitucion(FILE * orig, FILE * dest, FILE * dicci, char ultchar)
{
    if (!(ultchar ==' ' || ultchar =='.' || ultchar ==',' || ultchar =='(' || ultchar
==')' || ultchar ==':' || ultchar =='\n'))
       return false;
    char pal_dicci[1440];
    char ultimochar;
    int posEncontrada = 0;
    char palabra[1440];
    int posCursorInicial = ftell(orig);
    bool quedan = true;

    fseek(orig,-1,SEEK_CUR);
    fgets(palabra,1440,orig);

    fseek(orig,posCursorInicial,SEEK_SET);
    fseek(dicci,0,SEEK_SET);

    while(quedan)
    {
       if (fgets(pal_dicci,1440,dicci)) // Vamos comprobando linea a linea del fichero.
       {
           if (tolower(pal_dicci[0]) == tolower(palabra[0])) // Si las palabras empiezan
igual
           {
              for (int i=1; i <= strlen(pal_dicci)-1; i++)
```

```
            {
                if (pal_dicci[i] == '>' && (palabra[i]==' ' || palabra[i]=='.' ||
palabra[i]==',' || palabra[i]==')' || palabra[i]=='\n'))
                { //si han coincidido todas las letras hasta encontrar '>' significa
que la palabra está contemplada
                  for (int j=i+1; j <= strlen(pal_dicci)-2; j++)
                  {
                      fputc(pal_dicci[j],dest); //así que se escribe lo que haya en la
parte derecha del '>'
                  }
                  fseek(orig,i-1,SEEK_CUR);
                  return true;
                }
                else
                {
                  if (tolower(palabra[i]) != tolower(pal_dicci[i])) //cuando una letra
no coincide entre las cadenas, se acaba con esa linea
                      break;
                }
            }
          }
      }
      else
      {
          quedan = false;
      }
    }

    return false;
}

//-----------------------------------------------
//Función que lee el fichero de noticias final.
//-----------------------------------------------
void LeerNoticias()
{
    //primero se convierte la codificación UTF-8 a ISO-8859-1 para que se pronuncien
tildes y ñ's entre otros
    system ("iconv --from-code=UTF-8 --to-code=ISO-8859-1 ./Archiaux/textovozRef.txt
> ./Archiaux/textovozISO8859.txt -c -s");

    FILE * fichero;
    char linea[1440];
    fichero = fopen("./Archiaux/textovozISO8859.txt","r");

    bool salir = false;

    do{
       festival_eval_command("(voice_JuntaDeAndalucia_es_pa_diphone)");
       for (int i=0; i<=1; i++)
       {
           if (fgets(linea,1440,fichero) == NULL)
              salir = true;
           else
              festival_say_text(linea);
       }
       sleep(1);
       festival_eval_command("(voice_JuntaDeAndalucia_es_sf_diphone)");
       for (int i=0; i<=1; i++)
       {
           if (fgets(linea,1440,fichero) == NULL)
              salir = true;
           else
              festival_say_text(linea);
       }
       sleep(1);
```

```
    } while(!salir);

    festival_wait_for_spooler();
}

//-------------------------------------------------------------------------
//Función que lee el texto indicado previa conversión al charset ISO-8859-1.
//-------------------------------------------------------------------------
void Decir(string eltexto) // Función para leer con la codificación adecuada la cadena
de texto suministrada.
{
    char *texto = strdup(eltexto.c_str());
    FILE * archiaux;

    archiaux = fopen("./Archiaux/archiaux","w");
    fputs(texto,archiaux);
    fflush(archiaux);
    fclose(archiaux);
    system ("iconv --from-code=UTF-8 --to-code=ISO-8859-1 ./Archiaux/archiaux >
./Archiaux/archiauxISO8859 -c -s");
    archiaux = fopen("./Archiaux/archiauxISO8859","r");
    system("rm -f archiaux");
    fgets(texto,1440,archiaux);
    festival_eval_command("(voice_el_diphone)");
    festival_say_text(texto);
    fclose(archiaux);
    system("rm -f ./Archiaux/archiauxISO8859");
    festival_wait_for_spooler();
}

//-----------------------------------------------------------------------------
//Función que devuelve la última tecla pulsada o fuerza la salida si ésta era Escape.
//-----------------------------------------------------------------------------
bool ComprobarTeclado()
{
    char tecla = UltimaTeclaPulsada();
    if (tecla == ' ')
    {
       return true;
    }
    if (tecla == 27)
    {
       cout << "\n\nGracias por usar nuestro programa. Vuelva pronto.\n\n";
       Decir("Gracias por usar nuestro programa. Vuelva pronto");
       exit(1);
    }
    return false;
}

//-------------------------------------------
//Función que devuelve la última tecla pulsada.
//-------------------------------------------
char UltimaTeclaPulsada() // Esta es una versión modificada del Kbhit para linux, para
que devuvelva la última tecla pulsada
{
  struct termios oldt, newt;
  int ch;
  int oldf;

  tcgetattr(STDIN_FILENO, &oldt);
  newt = oldt;
  newt.c_lflag &= ~(ICANON | ECHO);
  tcsetattr(STDIN_FILENO, TCSANOW, &newt);
  oldf = fcntl(STDIN_FILENO, F_GETFL, 0);
  fcntl(STDIN_FILENO, F_SETFL, oldf | O_NONBLOCK);
```

```
  ch = getchar();

  tcsetattr(STDIN_FILENO, TCSANOW, &oldt);
  fcntl(STDIN_FILENO, F_SETFL, oldf);

    if(ch != EOF)
    {
       ungetc(ch, stdin);
       return ch;
    }

  return NULL;
}

//----------------------------------------
//Función para vaciar el buffer de entrada.
//----------------------------------------
void VaciarBufferTeclado()
{
    setvbuf(stdin, NULL, _IONBF,0); // Al deshabilitar la entrada estandar se vacía el
buffer,
    setvbuf(stdin, NULL, _IOFBF,0); //tras lo cual volvemos a habilitarla.
}

//-------------------------------------------------------------------------------------
//Función que comprueba si las voces a utilizar están instaladas y avisa en caso
contrario.
//-------------------------------------------------------------------------------------
void ComprobarVoces()
{
    DIR* directorio;
    bool estavoz1 = false, estavoz2 = false, estavoz3 = false;
    char dirvoz1[72] =
"/usr/share/festival/voices/spanish/JuntaDeAndalucia_es_pa_diphone"; //voz masculina de
la Junta de Andalucía
    char dirvoz2[72] =
"/usr/share/festival/voices/spanish/JuntaDeAndalucia_es_sf_diphone"; //voz femenina de
la Junta de Andalucía
    char dirvoz3[48] = "/usr/share/festival/voices/spanish/el_diphone"; //voz masculina
estándar
    char carpetavoz1[32] = "JuntaDeAndalucia_es_pa_diphone";
    char carpetavoz2[32] = "JuntaDeAndalucia_es_sf_diphone";
    char carpetavoz3[12] = "el_diphone";

    if ((directorio = opendir(dirvoz1)) != NULL)
       estavoz1 = true;

    if ((directorio = opendir(dirvoz2)) != NULL)
       estavoz2 = true;

    if ((directorio = opendir(dirvoz3)) != NULL)
       estavoz3 = true;

    if (!estavoz1 || !estavoz2 || !estavoz3)
    {
       printf("Algunas voces utilizadas no están instaladas. Esto puede afectar
sensiblemente la calidad de la experiencia:\n");
       if (!estavoz1)
           printf("%s\n",carpetavoz1);
       if (!estavoz2)
           printf("%s\n",carpetavoz2);
       if (!estavoz3)
           printf("%s\n",carpetavoz3);
       fflush(stdout);
       Decir("algunas voces utilizadas no están instaladas. Esto puede afectar
sensiblemente la calidad de la experiencia");
```

```
    }
}


//-------------------------------------------------------------------------------------
//Función que carga el controlador de sonido indicado en el fichero configaudio, que en
caso de no existir lo crea.
//-------------------------------------------------------------------------------------
void InicializarControlador()
{
    FILE * archiconfig;
    char codigodriver;

    archiconfig = fopen("configaudio","r");
    if (archiconfig == NULL)
    {
       codigodriver = BuscaControlador();
       if (codigodriver == '0')
       {
           cout << "\n¡No se encuentra un controlador de sonido adecuado!\nSaliendo de la
aplicación...\n\n";
           exit(1);
       }
       archiconfig = fopen("configaudio","w");
       fputc(codigodriver,archiconfig);
       fclose(archiconfig);
    }
    else
    {
       codigodriver = fgetc(archiconfig);
       fclose(archiconfig);
       switch (codigodriver)
       {
           case '1': festival_eval_command("(Parameter.set 'Audio_Command \"aplay -D
plug:dmix -q -c 1 -t raw -f S16_BE -r $SR $FILE\")"); break;
           case '2': festival_eval_command("(Parameter.set 'Audio_Command \"esdaudio
$FILE\")"); break;
           case '3': festival_eval_command("(Parameter.set 'Audio_Command \"sun16audio
$FILE\")"); break;
           case '4': festival_eval_command("(Parameter.set 'Audio_Command \"sunaudio
$FILE\")"); break;
           case '5': festival_eval_command("(Parameter.set
'Audio_Command \"freebsd16audio $FILE\")"); break;
           case '6': festival_eval_command("(Parameter.set 'Audio_Command \"linux16audio
$FILE\")"); break;
       }
       festival_eval_command("(Parameter.set 'Audio_Method 'Audio_Command)");
       festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
    }
    system("clear");
}

//-------------------------------------------------------------------------------------
//Función que se encarga de ir probando los controladores uno a uno hasta que encuentra
uno válido y lo guarda.
//-------------------------------------------------------------------------------------
char BuscaControlador()
{
    time_t Tantes, Tdespues;

    system("clear");
    cout << "Detectada primera ejecución. Buscando controlador de audio adecuado...";
    fflush(stdout);
    sleep(3);

    festival_eval_command("(Parameter.set 'Audio_Command \"aplay -D plug:dmix -q -c 1 -t
```

```
raw -f S16_BE -r $SR $FILE\")");
    festival_eval_command("(Parameter.set 'Audio_Method 'Audio_Command)");
    festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
    time(&Tantes);
    Decir("detectado el controlador ALSA");
    system("clear");
    time(&Tdespues);
    if (difftime(Tdespues,Tantes) >= 2)
       return '1';

    festival_eval_command("(Parameter.set 'Audio_Method 'esdaudio)");
    festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
    time(&Tantes);
    Decir("detectado el controlador E S D.");
    system("clear");
    time(&Tdespues);
    if (difftime(Tdespues,Tantes) >= 2)
       return '2';

    festival_eval_command("(Parameter.set 'Audio_Method 'sun16audio)");
    festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
    time(&Tantes);
    Decir("detectado el controlador san audio 16 bits");
    system("clear");
    time(&Tdespues);
    if (difftime(Tdespues,Tantes) >= 2)
       return '3';

    festival_eval_command("(Parameter.set 'Audio_Method 'sunaudio)");
    festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
    time(&Tantes);
    Decir("detectado el controlador san audio");
    system("clear");
    time(&Tdespues);
    if (difftime(Tdespues,Tantes) >= 2)
       return '4';

    festival_eval_command("(Parameter.set 'Audio_Method 'freebsd16audio)");
    festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
    time(&Tantes);
    Decir("detectado el controlador frii B S D audio 16 bits");
    system("clear");
    time(&Tdespues);
    if (difftime(Tdespues,Tantes) >= 2)
       return '5';

    festival_eval_command("(Parameter.set 'Audio_Method 'linux16audio)");
    festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
    time(&Tantes);
    Decir("detectado el controlador Linux audio 16 bits");
    system("clear");
    time(&Tdespues);
    if (difftime(Tdespues,Tantes) >= 2)
       return '6';

    return '0';
}
```

5.3_ WEB-2-VOICE

5.3.1_ PROPÓSITO

El objetivo de esta aplicación es esencialmente el mismo que el de la anterior “Web-a-Voz”, con la diferencia de que en esta ocasión se trata de acceder a webs en inglés. Así pues, se pretende ofrecer una manera alternativa de acceder a la información escrita disponible en ficheros RSS en las webs de prensa digital, utilizándolos para extraer las noticias y leerlas, abriendo así un abanico informativo inaccesible por personas con deficiencias visuales y, por qué no, ofreciendo también un medio alternativo de información para el resto de personas.

En esta ocasión no se presentarán opciones ni mensajes más allá de los iniciales de comprobación de controladores de sonido, ya que no se esperará ninguna acción del usuario. Esto se debe a que la versión actual siempre accederá directamente a la web de noticias Slashdot (http://slashdot.org/), pues se ha considerado que los mecanismos de selección de páginas están más que suficientemente vistos con las cinco alternativas de la versión de “Web-A-Voz” vista recientemente, evitando reexplicar código y centrándonos así en un único ejemplo que focalice la atención en los nuevos mecanismos que utilizaremos en esta ocasión para tratar las páginas.

5.3.2_ PRERREQUISITOS

Una vez más, para asegurar la correcta compilación y ejecución de la aplicación “Web-2-Voice”, mostraremos una lista de las instalaciones previas de software necesarias, así como las versiones utilizadas en nuestro equipo durante las pruebas y la programación de la aplicación. De nuevo nombraremos todos los paquetes y programas necesarios sin tener en cuenta que se hayan podido instalar anteriormente para probar los ejemplos de secciones anteriores o alguna de las otras aplicaciones que componen este proyecto.

Festival y su API de programación
El motor de síntesis del habla Festival, las cabeceras y funciones para su integración en aplicaciones.
Versión utilizada: 1:2.0.95-beta-2ubuntu1
Orden de consola para instalación desde el repositorio:

apt-get install festival festival-dev

Wget
La utilidad de red para la obtención de archivos de la web desde el shell. En nuestra distribución 10.10 de Ubuntu ya se encontraba instalada.
Versión utilizada: 1.12-1.1ubuntu3
Orden de consola para instalación desde el repositorio:

apt-get install wget

Xalan development kit
Librerías necesarias para integrar el procesador de transformaciones Xalan (visto en la sección 4.3) en las aplicaciones C++.
Versión utilizada: 1.10-4
Orden de consola para instalación desde el repositorio:

apt-get install libxalan110-dev

G++
El compilador GNU de C++.
Versión utilizada: 4:4.4.4-1ubuntu2
Orden de consola para instalación desde el repositorio:

apt-get install g++

Por último recordar que el Sistema Operativo en el que se ha creado y probado la aplicación es:
Ubuntu
Versión: 10.10

Como hemos venido advirtiendo anteriormente, en caso de disponer de versiones de software o sistema operativo distintos a los recién indicados no podemos asegurar que el funcionamiento sea idéntico al descrito en la siguiente subsección.

5.3.3_ FUNCIONAMIENTO

La aplicación completa está disponible en la carpeta "APLICACIONES/Web2Voice" del anexo.

En primer lugar procedamos a compilar la aplicación. Una vez más, basta con situarnos en el directorio correspondiente a ésta desde la consola y ejecutar la herramienta de generación de código introduciendo "make", aprovechando que en nuestras aplicaciones

hay incluído un Makefile que contiene la línea necesaria para la compilación:

g++ web2voice.cc -I/usr/lib -l Festival
-I/usr/include/festival -I/usr/include/speech_tools/
-leststring -lestbase -lestools -o web2voice

Si todo ha ido bien, ya tendremos el ejecutable "web2voice" listo para ejecutar en el directorio principal. Pero antes vamos a repasar el contenido inicial de la carpeta del programa.

Carpetas:

Archiaux: Contendrá ficheros auxiliares correspondientes a los refinamientos y conversiones de charsets del documento a leer.

Archiweb: Contendrá los ficheros RSS descargados de la webs seleccionadas para su lectura.

Ficheros:

web2voice: Ejecutable de la aplicación.

web2voice.cc: Fichero fuente de la aplicación. En la siguiente sección explicaremos cada una de las funciones que contiene.

Makefile: Fichero con la orden para generar la aplicación a través de la herramienta de generación "make".

XML2SABLE.xsl: Hoja de transformaciones XSL para extraer la información deseada del RSS de slashdot.org y completarla con etiquetas SABLE para crear así automáticamente un fichero de este tipo perfectamente interpretable por Festival. Veamos qué contiene esta hoja:

```
<?xml version="1.0"?>

<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:foo="http://purl.org/rss/1.0/">
    <xsl:output method="xml"/>

    <xsl:template match="/">
<SABLE>
<SPEAKER NAME="male1">
<VOLUME LEVEL="90%">
        <xsl:apply-templates select="/rdf:RDF/foo:item"/>
```

```
</VOLUME>
</SPEAKER>
</SABLE>
    </xsl:template>

    <xsl:template match="/rdf:RDF/foo:item">
<PITCH RANGE="60%">
<RATE SPEED="-10%">
            <xsl:value-of select="foo:title"/>.
</RATE>
</PITCH>
<BREAK/>
                    <xsl:value-of select="foo:description/foo:text"
disable-output-escaping="yes" />.
<BREAK/>
    </xsl:template>

</xsl:stylesheet>
```

NOTA: Si no se ha hecho todavía, recomendamos encarecidamente repasar la sección 4.3, que ayudará a comprender mejor Xalan y la utilidad y funcionamiento de estas hojas XSL.

Repasemos el contenido del fichero. Inicialmente, se incluyen unas líneas cuya finalidad es indicar que el fichero que se tratará con esta hoja es un RSS, con estructura XML. A continuación se accede a la etiqueta raíz con `<xsl:template match="/">` y se procede a incluir etiquetas de SABLE como la de apertura `<SABLE>`, la de utilización de la voz inglesa masculina `<SPEAKER NAME="male1">` y la de cambio de volumen `<VOLUME LEVEL="90%">`, tras lo cual con `<xsl:apply-templates select="/rdf:RDF/foo:item"/>` se procede a buscar coincidencias con las etiquetas que se especifican más adelante que se encuentren dentro de todas las marcas <item> contenidas en <rdf> (el nodo principal). Entonces se cierran las etiquetas de SABLE abiertas anteriormente, que englobarán todo el texto resultante, y posteriormente se especificará el comportamiento de las transformaciones cuando se llegue a las etiquetas contenidas en la anteriormente indicada:

-En primer lugar se contemplan las etiquetas de título con `<xsl:value-of select="foo:title"/>` las cuales antes de escribir su contenido estarán encerradas por etiquetas `<PITCH RANGE="60%">` y `<RATE SPEED="-10%">` para dar más entonación y reducir la velocidad de lectura de los títulos para darles más importancia.

-En segundo lugar, se contemplan las etiquetas `<text>` contenidas en `<description>` mediante `<xsl:value-of select="foo:description/foo:text" disable-output-escaping="yes" />`, en esta ocasión sin etiquetas SABLE que alteren el texto resultante (cuerpo de la noticia). Las etiquetas <text> no aperecen originalmente en el archivo de noticias obtenido, sino que es un

añadido que incluímos en un pequeño preprocesamiento del texto, útil para aislar el texto deseado y esquivar símbolos y etiquetas que no cumplen extrictamente las especificaciones XML, como < o > que se utilizan para mejorar la presentación en lectores basados en HTTP como Atom.

Al aplicar esta hoja con los mecanismos comentados obtenemos una hoja SABLE correctamente formada, lista para ser interpretada por Festival. En la carpeta "EJEMPLOS/SABLE/XML-XSL-SABLE" hay disponible un ejemplo real de transformación de un fichero de noticias RSS, transformado con esta hoja XSL, dando como resultado el fichero SABLE (a los que hemos llamado "Origen.xml", "Transformacion.xsl" y "Resultado.sable" respectivamente). Las etiquetas SABLE que hemos añadido en la hoja de transformaciones son un mero ejemplo para mostrar cómo se puede alterar la lectura con ellas, habiendo probablemente muchas y mejores posibilidades. No obstante, la imprevisibilidad en la forma y los contenidos de las noticias hacen que la aplicación correcta de muchas de las etiquetas sea imposible en procesos automáticos como este.

Funcionamiento paso a paso:

En primer lugar ejecutamos el programa escribiendo "./web2voice" en la consola. Si es la primera vez que lo ejecutamos, un mensaje nos alertará de esto y se procederá a buscar y almacenar el controlador de sonido adecuado. En esta ocasión no se ha incluído la comprobación de voces, pues se utilizará la voz masculina inglesa, disponible con la propia instalación de Festival.

Inmediatamente después, se procederá directamente a la lectura de las noticias obtenidas desde la web de Slashdot (http://slashdot.org/), sin requerir acción ninguna por motivos de simplicidad como indicábamos en la la sección 5.3.1.

Descripción de las funciones

Vamos a describir el funcionamiento de cada una de las funciones implicadas en la aplicación, las cuales están disponibles al completo en la siguiente sección (5.3.4). Aunque en menor medida, de nuevo algunas funciones son reutilizadas, pero volveremos a explicarlas para mantener la cohesión del conjunto de todas ellas. Si se han seguido las descripciones de aplicaciones anteriores, en esta quizás se eche en falta la función RefinarTexto, que no se ha incluído porque gracias a las transformaciones XSL se obtiene un fichero muy "limpio", además de que según las pruebas realizadas la voz en inglés está más perfeccionada para ignorar caracteres no alfanuméricos y anomalías.

(1) → int Main(int argc, char **argv)

Es la función principal, el punto de entrada de nuestra aplicación. En ella se realizan secuencialmente las siguientes acciones:
1- Inicializar Festival como de costumbre.
2- Llamar a la función **InicializarControlador→(2)**.
3- Descargar el fichero de noticias con **ObtenerPagina→(4).**
4- Modificar ligeramente el fichero con **ModificarPagina→(5)**.
5- Realizar la conversión XML a SABLE con **TransformarPagina→(8)**.
6- Proceder a leer el fichero resultante con **LeerPagina→(9)**.

(2) → void InicializarControlador()

Es la función encargada de cargar el controlador de sonido adecuado para nuestro equipo. Lo que hace es comprobar si existe un fichero llamado "configaudio" en el directorio principal de la aplicación. Si no existe, llama a la función **BuscaControlador→(3)**, que devuelve un código que si es distinto de cero se almacena en el fichero "configaudio" que crea en ese mismo instante, mientras que si es cero se alerta de que no se encuentra un controlador adecuado y se finaliza la aplicación. Después utiliza ese código (entre 1 y 6) para ejecutar una instrucción de Festival que cambia el controlador de sonido a utilizar correspondiente a dicho código. Finalmente limpia la pantalla.

(3) → char BuscaControlador()
-Devuelve: el código del controlador adecuado entre 1 y 6, o un 0 si no encuentra ninguno.

Esta función no hace otra cosa que ir cargando uno a uno los seis controladores de sonido que conocemos, intentando reproducir para cada uno de ellos una frase afirmando que se ha encontrado dicho controlador. En esta aplicación ya no necesitamos la función **Decir** que utilizábamos en las otras dos, debido a que para leer textos en inglés es suficiente leer las palabras con la codificación por defecto UTF-8 con la función *festival_say_text*. La aplicación cronometra esos intentos de lectura de cada frase y comprueba el tiempo en segundos obtenido; Sólo cuando ese tiempo sea igual o mayor a dos segundos, significará que la frase efectivamente se ha leído, por lo que se considera válido el último controlador cargado y se devuelve su código correspondiente. Si ningún controlador ha sido válido, el valor de retorno será entonces cero.
Hemos tenido que idear una estrategia de detección así de

curiosa, porque las funciones de Festival no se han comportado como esperábamos: la función que cambia de controlador devuelve 1 siempre independientemente de si el controlador es válido o no, y la función para leer un texto devuelve 1 tanto si consigue leer como si no.

(4)→ void ObtenerPagina()

En esta función, primero se elimina la anterior descarga del fichero "slashdot" (nombre del fichero RSS que ofrece Slashdot.org), y a continuación se procede a descargar la última versión del mismo de la forma habitual con la utilidad externa *wget*.

(5)→ void ModificarPagina()

El objetivo de esta función es crear, a partir del fichero "slashdot" descargado, un fichero llamado "paginamod.xml" idéntico al anterior con la única diferencia de la inclusión de etiquetas *<text>* y *</text>* encerrando la información legible de la descripción de las noticias, tal como comentábamos en la descripción del fichero "XML2SABLE.xsl" en esta misma sección. Para ello, lo que hacemos es ir recorriendo el fichero poniendo la etiqueta de apertura *<text>* cuando encontramos la etiqueta *<description>*, o la etiqueta de cierre *</text>* cuando encontramos *</description>* o *<p>*. Para comprobar esas etiquetas y cadenas usamos las habituales **ProxCaracteres→(6)** y **AntCaracteres→(7)**.

(6) → bool ProxCaracteres(FILE * fdesc, string laetiqueta)
-Recibe: el descriptor del fichero de lectura y el texto a comprobar.
-Devuelve: 'True' cuando la cadena recibida se encuentra a partir de la posición actual del cabezal de lectura en el fichero, o 'False' en caso contrario.

Incialmente se guarda la posición del puntero de lectura del fichero, y se procede a comparar los siguientes caracteres uno a uno con los del string suministrado. Si alguno de ellos no coincide, se devuelve 'False' y se devuelve el puntero de lectura a su posición inicial, mientras que si todas las comparaciones tienen éxito se devuelve 'True'.

(7) → bool AntCaracteres(FILE * fdesc, string laetiqueta)
-Recibe: el descriptor del fichero de lectura y el texto a comprobar.

-Devuelve: 'True' cuando la cadena recibida se encuentra a partir de la posición actual del cabezal de lectura del fichero, o 'False' en caso contrario.

De forma análoga a la función anterior, ésta almacena la posición actual del puntero de lectura y procede a comparar carácter a carácter, pero esta vez hacia atrás. Si alguna comparación falla devuelve 'False' y si todas tienen éxito devuelve 'True'. En cualquiera de los dos casos devuelve el puntero de lectura a la posición almacenada inicialmente.

(8)→ void TransformarPagina()

En esta función se realiza la tranformación del archivo de noticias XML modificado, en un archivo ".sable" utilzando la hoja de transformación "XML2SABLE.xsl". Esto se consigue a través de las funciones de la API C++ de Xalan que instalamos en la sección 5.3.2, que utilizadas convenientemente como se explica en la web de Apache[64] y otros tutoriales[65] sobre Xalan, realizan la misma conversión que se consigue utilizando la utilidad de consola (explicada en la sección 4.3). Para que funcione, préviamente se deben incluir las siguientes referencias e inicializaciones antes de la función *main*:

```
#include <xalanc/Include/PlatformDefinitions.hpp>
#include <xercesc/util/PlatformUtils.hpp>
#include <xalanc/XalanTransformer/XalanTransformer.hpp>
#include <xalanc/XalanTransformer/XalanCAPI.h>

XALAN_USING_XERCES(XMLPlatformUtils)
XALAN_USING_XALAN(XalanTransformer)
XALAN_USING_XALAN(XSLTInputSource)
XALAN_USING_XALAN(XSLTResultTarget)
```

Ya centrados en la función, primero se inicializa Xalan. A continuación se declara un objeto "XalanTransformer" al que, previo almacenamiento de las rutas de los tres ficheros implicados (el XML de origen, el XSL de la transformación y el SABLE resultante), realiza la transformación esperada a través del método "transform", al que se le pasan como argumento las rutas citadas. Tras esto ya estará disponible el fichero "pagina.sable" y sólo queda finalizar Xalan.

(9)→ void LeerPagina()

En esta última función, primero se comunica que se va a proceder a leer las noticias. A continuación, se comienza a leer directamente el fichero "pagina.sable" que obtuvimos en la función

anterior, con los cambios en entonación, velocidad y volumen que habíamos introducido a través de la hoja XSL.

5.3.4_ CÓDIGO FUENTE

A continuación se expone el código fuente de la aplicación "Web-2-Voice" al completo:

```
#include <stdio.h>
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <festival.h>
#include <sstream>
#include <string>
#include <termios.h>
#include <fcntl.h>

#include <xalanc/Include/PlatformDefinitions.hpp>
#include <xercesc/util/PlatformUtils.hpp>
#include <xalanc/XalanTransformer/XalanTransformer.hpp>
#include <xalanc/XalanTransformer/XalanCAPI.h>

XALAN_USING_XERCES(XMLPlatformUtils)
XALAN_USING_XALAN(XalanTransformer)
XALAN_USING_XALAN(XSLTInputSource)
XALAN_USING_XALAN(XSLTResultTarget)

void ObtenerPagina(), InicializarControlador(), ModificarPagina(), TransformarPagina(),
LeerPagina();
bool AntCaracteres(FILE*,string), ProxCaracteres(FILE*,string);
char BuscaControlador();

int heap_size = 210000;
int load_init_files = 1;

int main(int argc, char **argv)
{
    festival_initialize(load_init_files,heap_size);

    festival_eval_command("(voice_kal_diphone)");   //Carga la voz por defecto de
Festival.

    InicializarControlador();    //Cargar el controlador de sonido adecuado

    ObtenerPagina();             //Descargar el RSS de slashdot.org

    ModificarPagina();           //Meter las etiquetas <text> para reconocer más
fácilmente las noticias
    TransformarPagina();  //XML ---XSL--> SABLE

    LeerPagina();         //Reproducir el aschivo SABLE resultante
}

//-------------------------------------------------
//Función para obtener el fichero RSS de slashdot.
//-------------------------------------------------
void ObtenerPagina()
{
    remove("./Archiweb/slashdot");
    system("wget -i -D http://rss.slashdot.org/Slashdot/slashdot -o ./Archiaux/log.txt
--directory-prefix=Archiweb");
```

```
}

//------------------------------------------------------------------------
//Función que realiza las modificaciones necesarias en la página descargada.
//------------------------------------------------------------------------
void ModificarPagina()
{
    FILE * forigen;
    FILE * fdestino;
    char caracter;
    bool abiertaText = false;    //indicará si se ha puesto una etiqueta de <text> y
estamos en espera del cierre </text>

    forigen = fopen("./Archiweb/slashdot","r");

    if (forigen != NULL)
    {
       fdestino = fopen("./Archiaux/paginamod.xml","w");
       caracter = fgetc(forigen);
       do{
           if (ProxCaracteres(forigen,"&amp;mdash; "))     //linea que elimina los
&amp;mdash; que aparecen en las noticias
           {
              fputc(' ',fdestino);
           }
           else if (ProxCaracteres(forigen,"&amp;mdash;")) //y aqui lo mismo pero
poniendo un espacio si no lo tenía
           {
              fputc(' ',fdestino);
           }

           else if (!abiertaText)
           {
              if (AntCaracteres(forigen,"<description>"))   //Cuando se encuentra una
etiqueta <description> se coloca la etiqueta <text>
              {
                  fputs("<text>",fdestino);
                  abiertaText = true;
              }
                  fputc(caracter,fdestino);
           }
           else if (abiertaText)
           {
              if (ProxCaracteres(forigen,"&lt;p&gt;"))      //y cuando se encuentra
&lt;p&gt;
              {
                  fputs("</text>&lt;p&gt;",fdestino);
                  abiertaText = false;
              }
              else if (ProxCaracteres(forigen,"</description>"))  //o </description>
              {
                  fputs("</text></description>",fdestino);        //se añade </text>
antes (se cierra)
                  abiertaText = false;
              }
              else
                  fputc(caracter,fdestino);
           }
           caracter = fgetc(forigen);
       } while(caracter != EOF); //hasta encontrar el fin de linea.

       fflush(fdestino);
       fclose(forigen);
       fclose(fdestino);
    }
}
```

```
//---------------------------------------------------------------------
//Función que obtiene un archivo SABKE a partir del XML de las noticias.
//---------------------------------------------------------------------
void TransformarPagina()
{
    XMLPlatformUtils::Initialize();
    XalanTransformer::initialize();
    {
    XalanTransformer theXalanTransformer;
    const char* xmlIn = "./Archiaux/paginamod.xml";
    const char* xslIn = "XML2SABLE.xsl";
    const char* xmlOut = "./Archiaux/pagina.sable";
    int theResult = theXalanTransformer.transform(xmlIn,xslIn,xmlOut);
    }
    XalanTransformer::terminate();
    XMLPlatformUtils::Terminate();
    XalanTransformer::ICUCleanUp();
}

//-------------------------------------------------
//Función que procede a leer el fichero SABLE final.
//-------------------------------------------------
void LeerPagina()
{
    printf("Reading news...\n\n");
    fflush(stdout);
    festival_say_text("reading news");
    festival_say_file("./Archiaux/pagina.sable");
}

//-----------------------------------------------------------------------------------------
//Función que comprueba si en las anteriores posiciones al puntero del fichero indicado
se encuentra la cadena.
//-----------------------------------------------------------------------------------------
bool AntCaracteres(FILE * fdesc, string laetiqueta)
{
    char proxcaracter;
    char *etiqueta = strdup(laetiqueta.c_str());
    int posCursorInicial = ftell(fdesc);

    fseek(fdesc,-(strlen(etiqueta)+1),SEEK_CUR);
    for (int i=0; i <= strlen(etiqueta)-1; i++)
    {
       proxcaracter = fgetc(fdesc);
       if (proxcaracter != etiqueta[i])
       {
           fseek(fdesc,posCursorInicial,SEEK_SET);
           return false;
       }
    }
    fseek(fdesc,posCursorInicial,SEEK_SET);
    return true;
}

//-----------------------------------------------------------------------------------------
//Función que comprueba si en las siguientes posiciones al puntero del fichero indicado
se encuentra la cadena.
//-----------------------------------------------------------------------------------------
bool ProxCaracteres(FILE * fdesc, string laetiqueta)
{
    char proxcaracter;
    char *etiqueta = strdup(laetiqueta.c_str());
    int posCursorInicial = ftell(fdesc);

    fseek(fdesc,-1,SEEK_CUR);
    for (int i=0; i <= strlen(etiqueta)-1; i++)
```

```
    {
       proxcaracter = fgetc(fdesc);
       if (proxcaracter != etiqueta[i])
       {
           fseek(fdesc,posCursorInicial,SEEK_SET);
           return false;
       }
    }

    return true;
}

//-------------------------------------------------------------------------------------
//Función que carga el controlador de sonido indicado en el fichero configaudio, que en
caso de no existir lo crea.
//-------------------------------------------------------------------------------------
void InicializarControlador()
{
    FILE * archiconfig;
    char codigodriver;

    archiconfig = fopen("configaudio","r");
    if (archiconfig == NULL)
    {
       codigodriver = BuscaControlador();
       sleep(1);
       if (codigodriver == '0')
       {
           cout << "\nUnable to find a proper audio device!\nExiting aplication...\n\n";
           exit(1);
       }
       archiconfig = fopen("configaudio","w");
       fputc(codigodriver,archiconfig);
       fclose(archiconfig);
    }
    else
    {
       codigodriver = fgetc(archiconfig);
       fclose(archiconfig);
       switch (codigodriver)
       {
           case '1': festival_eval_command("(Parameter.set 'Audio_Command \"aplay -D
plug:dmix -q -c 1 -t raw -f S16_BE -r $SR $FILE\")"); break;
           case '2': festival_eval_command("(Parameter.set 'Audio_Command \"esdaudio
$FILE\")"); break;
           case '3': festival_eval_command("(Parameter.set 'Audio_Command \"sun16audio
$FILE\")"); break;
           case '4': festival_eval_command("(Parameter.set 'Audio_Command \"sunaudio
$FILE\")"); break;
           case '5': festival_eval_command("(Parameter.set
'Audio_Command \"freebsd16audio $FILE\")"); break;
           case '6': festival_eval_command("(Parameter.set 'Audio_Command \"linux16audio
$FILE\")"); break;
       }
       festival_eval_command("(Parameter.set 'Audio_Method 'Audio_Command)");
       festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
    }
    system("clear");
}

//-------------------------------------------------------------------------------------
//Función que se encarga de ir probando los controladores uno a uno hasta que encuentra
uno válido y lo guarda.
//-------------------------------------------------------------------------------------
char BuscaControlador()
{
```

```
    time_t Tantes, Tdespues;

    system("clear");
    cout << "First execution detected. Searching the proper audio device...";
    fflush(stdout);
    sleep(3);

    festival_eval_command("(Parameter.set 'Audio_Command \"aplay -D plug:dmix -q -c 1 -t
raw -f S16_BE -r $SR $FILE\")");
    festival_eval_command("(Parameter.set 'Audio_Method 'Audio_Command)");
    festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
    time(&Tantes);
    festival_say_text("ALSA controller detected");
    system("clear");
    time(&Tdespues);
    if (difftime(Tdespues,Tantes) >= 2)
       return '1';

    festival_eval_command("(Parameter.set 'Audio_Method 'esdaudio)");
    festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
    time(&Tantes);
    festival_say_text("E S D controller detected");
    system("clear");
    time(&Tdespues);
    if (difftime(Tdespues,Tantes) >= 2)
       return '2';

    festival_eval_command("(Parameter.set 'Audio_Method 'sun16audio)");
    festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
    time(&Tantes);
    festival_say_text("sun audio 16 bits controller detected");
    system("clear");
    time(&Tdespues);
    if (difftime(Tdespues,Tantes) >= 2)
       return '3';

    festival_eval_command("(Parameter.set 'Audio_Method 'sunaudio)");
    festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
    time(&Tantes);
    festival_say_text("sun audio controller detected");
    system("clear");
    time(&Tdespues);
    if (difftime(Tdespues,Tantes) >= 2)
       return '4';

    festival_eval_command("(Parameter.set 'Audio_Method 'freebsd16audio)");
    festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
    time(&Tantes);
    festival_say_text("free B S D audio 16 bits controller detected");
    system("clear");
    time(&Tdespues);
    if (difftime(Tdespues,Tantes) >= 2)
       return '5';

    festival_eval_command("(Parameter.set 'Audio_Method 'linux16audio)");
    festival_eval_command("(Parameter.set 'Audio_Required_Format 'snd)");
    time(&Tantes);
    festival_say_text("Linux audio 16 bits controller detected");
    system("clear");
    time(&Tdespues);
    if (difftime(Tdespues,Tantes) >= 2)
       return '6';

    return '0';
}
```

6_ CONCLUSIONES

Llegados a este punto, los ejemplos y aplicaciones mostrados a lo largo de la memoria habrán dado al lector una base general en lo referente al área de la síntesis de voz, tanto de sus ventajas y puntos positivos como de sus inconvenientes. Recojamos a continuación algunas de estas conclusiones a las que hemos llegado a través de la investigación y la experiencia.

POSITIVAS:

- Abre el acceso a un medio de información inaccesible de otra forma para personas con deficiencias visuales severas.
- Ofrece una alternativa al acceso de un medio que, siendo escrito, puede no ser adecuado en ciertas situaciones o entornos de trabajo.
- En principio, cualquier texto presente en la WWW es susceptible de ser tratado e interpretado.
- La existencia de las APIs para motores de síntesis hacen que la programación de aplicaciones TTS sea fácil e intuitiva.
- La reutilización de las funciones que vayamos desarrollando es potencialmente alta.
- La extensa comunidad implicada en este tema hace que exista una considerable cantidad de información y documentación, además de que constribuye a la aparición de nuevas voces en multitud de idiomas.

NEGATIVAS:

- Es un mundo que, al no ser tenido en cuenta por la gran inmensa mayoría de las webs, no se ofrecen facilidades de acceso debiendo hacer en ocasiones tediosos procesos de filtrado y corrección de ficheros pensados para otro fin.
- La imprevisibilidad en la forma y los contenidos de los ficheros obtenidos (como los RSS de noticias vistos) complica la automatización y restringe la posibilidad de colocación de las etiquetas de lenguajes de marcado a apenas unas pocas.
- Ninguna de las voces de Festival probadas por nosotros, aparte de las inglesas oficiales, soporta los lenguajes de marcado.
- Se necesita un formato de texto plano y con una codificación adecuada al lenguaje utilizado.
- Los motores no tienen la capacidad de identificar por sí sólos cuándo la lectura de ciertas palabras debe hacerse con las reglas de otros idiomas, necesitando así la invención de mecanismos como el diccionario de sustituciones propuesto por nosotros.

7_ TRABAJOS FUTUROS

Las aplicaciones desarrolladas a lo largo de este proyecto cumplen los objetivos para los que fueron pensadas, pero existen infinidad de ampliaciones y correcciones posibles que elevarían sus prestaciones. Aquí recogemos las que nos vienen en mente a día de hoy, clasificadas en generales y específicas a cada aplicación.

AMPLIACIONES Y MEJORAS GENERALES:

– Podría mejorarse la apariencia de cada aplicación mediante una GUI (interfaz gráfica), aunque la información y aspecto visual no ha sido en ningún momento una preocupación prioritaria.

– Las aplicaciones que acceden al fichero de sustituciones se ralentizarían exponencialmente conforme crezca la cantidad de sustituciones (líneas) contempladas en él. Para solucionarlo, sería una buena idea crear un array con 27 entradas, una por cada letra del abecedario, cada una de las cuales sea un puntero a una estructura que contenga dos variables enteras: una “inicio” y otra “final”, de manera que tras el mismo momento en que se inicia la aplicación, se realice un análisis del fichero de sustituciones y se vaya guardando para cada letra en qué línea comienzan las palabras que empiezan por ella y en qué linea finalizan. Así, cada vez que se busca una palabra para sustituir se haría estrictamente en las líneas en que hubiera palabras que empiecen por su letra inicial. De este modo, si no tenemos en cuenta el análisis de frecuencias de aparición de las letras en nuestro idioma y suponemos que todas aparecen con la misma probabilidad, tendríamos que la función de refinamiento del texto tardaría 1/27 de lo que se tarda sin indizar las entradas, o lo que es lo mismo un 3,7%. Por supuesto sería indispensable mantener las palabras del fichero de sustituciones ordenadas alfabéticamente, al menos por su letra inicial.

– Las aplicaciones en castellano necesitaban una conversión de la codificación de caracteres, que realizábamos mediante la utilidad Iconv, invocándola desde una orden externa de consola. Idealmente nuestras aplicaciones deberían evitar en la medida de lo posible las dependencias con órdenes externas, condicionadas a la instalación de otros softwares y disminuyendo la eficiencia de las aplicaciones resultantes. Para ello, convendría instalar y estudiar la librería *libc-bin*, disponible desde el repositorio de la forma habitual, que proporciona el conjunto de cabeceras y librerias necesarias para realizar las conversiones en nuestras aplicaciones C++.

– Podríamos ser más ambiciosos y extender la funcionalidad de nuestras aplicaciones al resto de herramientas de nuestro PC, como por ejemplo un lector del correo electrónico.

– Además de escuchar los archivos in situ, sería sencillo añadir a las aplicaciones la opción de guardar la síntesis resultante en un fichero de audio, lo cual permitiría poder escucharlos en dispositivos portátiles como los MP3 y no tener que estar cerca del PC. Técnicamente es algo tan simple como utilizar la orden *text2wave* en lugar de las habituales *festival_say_file* y *festival_say_text*.

AMPLIACIONES Y MEJORAS PARA E-NARRADOR:

- Guardar la posición al final de cada párrafo puede resultar insuficiente, pues en caso de pulsar Escape durante párrafos largos puede tocarnos esperar demasiado tiempo antes de que el párrafo acabe, guarde la posición y podamos irnos. Por ello quizá sería buena idea al guardar la posición hacerlo oración a oración (punto a punto) en lugar de párrafo a párrafo (línea a línea). Para ello bastaría con llevar un conteo de los puntos que se van encontrando en el texto.

- Cuando se finaliza la ejecución con control+C durante la lectura de un documento, en lugar de finalizar la ejecución de la aplicación, se interrumple la lectura del párrafo actual, por lo que se necesita un segundo control+C para finalizar la aplicación antes de que se comience a leer el párrafo siguiente. Esto es debido a que Festival se ejecuta “por encima” de la aplicación C++ y como consecuencia de la situación descrita puede ocurrir que el párrafo almacenado en la guía sea el siguiente al que debería ser. Se podría intentar solucionar esto identificando el PID del proceso de Festival, para que posteriormente mediante un manejador de la interrupción de Control+C se envíe una señal de terminación al proceso antes de finalizar la propia aplicación.

AMPLIACIONES Y MEJORAS PARA WEB-A-VOZ:

- Sería posible incluir un fichero de fuentes en el que el usuario pudiera añadir sus propias direcciones de noticias RSS favoritas, con lo cual pasaran a estar disponibles en el menú de la aplicación para su lectura. Se necesitaría unificar una gran función de refinamiento de textos genérica que contemplara el mayor número posible de “anomalías” habituales a corregir.

- Se podría llevar más allá el acceso a las noticias dando la posibilidad de consultar la noticia completa correspondiente al

avance que se esté escuchando cuando se pulse una tecla determinada. El proceso se prevee laborioso, pero básicamente consistiría en descargar el fichero al que enlaza la noticia del RSS (habitualmente una dirección web encerrada en etiquetas <link> o <a>) y diseñar un filtrado y refinamiento similares a los realizados aunque presumiblemente más costosos (evitando vídeos, imágenes, banners...).

AMPLIACIONES Y MEJORAS PARA WEB-2-VOICE:

- Se podría idear una interfaz que permita modificar en tiempo de ejecución los diferentes aspectos que soporta SABLE. Para ello, puesto que un fichero SABLE no se puede modificar una vez comienza a ser interpretado por Festival, lo que se debería hacer es ir creando pequeños ficheros SABLE que contengan cada uno una sola frase, y todas las etiquetas posibles para modificar la lectura encerrándola. Así, desde la interfaz se podrían alterar esos parámetros almacenados como variables en la aplicación, que serían suministradas como parámetros de las etiquetas SABLE de los pequeños ficheros que se van creando.

8_ BIBLIOGRAFÍA Y REFERENCIAS

(1) Festival: SABLE Markup (Alan W. Black)
http://www.cs.cmu.edu/~awb/festival_demos/sable.html
(página de Alan W Black, profesor de ingeniería informática en la Carnegie Mellon University).

(2) Festival Speech Synthesis System.
http://www.speech.cs.cmu.edu/festival/manual-1.4.1
(manual de Festival, a cargo de la facultad de informática de la Carnegie Mellon University).

(3) World Wide Web Consorcium (W3C).
http://www.w3.org/
(información sobre los estándares citados).

(4) Really Simple Syndication.
http://es.wikipedia.org/wiki/RSS
(información sobre RSS en la wikipedia).

(5) Prácticas de FSMM.
http://web-sisop.disca.upv.es/~fsmm/practiques/indexPractiquesFSMM.html
(prácticas 4 y 5 sobre XML y XSL de la asignatura FSMM del DISCA en la UPV).

(6) El ludismo en el siglo XIX
http://es.wikipedia.org/wiki/Ludismo
(artículo sobre el ludismo)

(7) IBM
http://www.ibm.com/es/es/
(página web de IBM España)

(8) Grabadora Phillips
http://www.philips.co.uk/c/dictation/voice-tracer-speech-to-text-lfh0625_00/prd?retailStoreStatus=true&country=GB&catalogType=CONSUMER&language=en&onlineStoreStatus=true&stores=true&proxybuster=74C1424BD3880D91EB182F9F280471C8.app101-drp2
(ejemplo de grabadora speech-to-text de Phillips en el mercado)

(9) Grabadoras Olympus
(http://www.olympus.es/voice/)
(Web de Olympus con varias grabadoras speeh-to-text)

(10) Stephen Hawking
http://es.wikipedia.org/wiki/Stephen_Hawking
(artículo de wikipedia sobre el astrofísico)

(11) Artículo de magazinedigital.com
http://www.magazinedigital.com/reportajes/ciencia/reportaje/cnt_id/3406/pageID/1
(artículo con una entrevista a Stephen Hawking que invluye una descripción de su sistema de comunicación)

(12) La prosodia
http://es.wikipedia.org/wiki/Prosodia
(artículo de la wikipedia sobre la prosodia)

(13) Voice Extensible Markup Language
http://www.w3.org/TR/voicexml21/
(Especificaciones del W3C sobre voiceXML)

(14) Voice browser
http://es.wikipedia.org/wiki/Navegador_web
(breve artículo de la wikipedia sobre los navegadores de voz)

(15) HyperText Markup Language
http://es.wikipedia.org/wiki/HTML
(artículo de la wikipedia sobre el lenguaje HTML)

(16) OpenVXI
http://www.speech.cs.cmu.edu/openvxi/index.html
(página de la Carnegie Mellon University sobre OpenVXI)

(17) JvoiceXML
http://jvoicexml.sourceforge.net/
(página del proyecto JvoiceXML en sourceforge.net)

(18) PublicVoiceXML
http://publicvoicexml.sourceforge.net/
(sitio oficial de PublicVoiceXML en sourceforge.net)

(19) Java Speech Markup Language
http://www.w3.org/TR/jsml/
(especificaciones del W3C sobre JSML)

(20) Speech Synteshis Markup Language
http://www.w3.org/TR/speech-synthesis/
(especificaciones del W3C sobre SSML)

(21) Skunk Template Markup Language
http://www.bell-labs.com/project/tts/stml-notes.html
(especificaciones de STML en la web de Bell-labs)

(22) SpeechSyntehis
http://en.wikipedia.org/wiki/Speech_synthesis#Concatenative_synthesis
(artículo de la wikipedia con descripciones y clasificaciones de la síntesis)

(23) Formante (en inglés, Formant)
http://es.wikipedia.org/wiki/Formante
(definición de "formante" en la wikipedia)

(24) Pitch
http://www.bodylanguageuniversity.com/public/203.cfm
(artículo que describe el pitch)

(25) Sistemas embebidos
http://www.idose.es/sistemas-embebidos
(definición de sistema embebido)

(26) Verbio
http://www.verbio.com/webverbio3/html/productes.php?PHPSESSID=e5a0b37c87a637a6e9c9d1cbafbca2fa
(sitio oficial de Verbio TTS)

(27) ASR: "Automatic Speech Recognition" o "Reconocimiento automático del habla", terminología equivalente a STT "Speech-To-Text" o "Habla-A-Texto".

(28) Free Text-To-Speech
http://freetts.sourceforge.net/docs/index.php#what_is_freetts
(especificaciones y descargas de Free TTS)

(29) Flite
http://www.speech.cs.cmu.edu/flite/
(descripción de Flite en la Carnegie Mellon University)

(30) Festvox Project
http://festvox.org/
(página oficial del proyecto Festvox)

(31) MBRola
http://tcts.fpms.ac.be/synthesis/
(página oficial de MBRola con especificaciones y ejemplos)

(32) Java Speech Api
http://java.sun.com/products/java-media/speech/
(página de Sun con la documentación de JSAPI)

(33) Loquendo
http://www.loquendo.com/es/productos/sintetizador-de-voz/
(página del software TTS de Loquendo)

(34) GNUSpeech
http://www.gnu.org/software/gnuspeech/#Whatis
(especificaciones y documentación de GNUSpeech)

(35) Nuance Realspeak
http://australia.nuance.com/realspeak/
(página de Nuance Realspeak)

(36) eSpeak
http://espeak.sourceforge.net/
(descripción y especificaciones de eSpeak)

(37) SodelsCot
http://www.sodels.com/
(web oficial de SodelsCot)

(38) Lernout & Hauspie
http://www.worldlingo.com/ma/enwiki/es/Lernout_%26_Hauspie
(orígenes de Lernout & Hauspie)

(39) NeoSpeech Voice Text
http://www.neospeech.com/
(web de NeoSpeech Voice Text)

(40) Fonix TTS
http://www.fonixspeech.com/tts.php
(web de Fonix TTS)

(41) the DAISY project
http://www.daisy.org/
(web del proyecto DAISY)

(42) the VOICES project
http://www.webfoundation.org/2011/01/new-project-voices-voice-based-community-centric-mobile-services-for-social-development/
(artículo sobre VOICES en la web de la World Wide Web Foundation)

(43) Tim Berners-Lee
http://www.w3.org/People/Berners-Lee/
(web personal de Tim Berners-Lee)

(44) World Wide Web Foundation
http://www.webfoundation.org/
(página principal de la World Wide Web Foundation)

(45) Asterisk Project
http://www.asterisk.org/
(web oficial del proyecto Asterisk)

(46) Interactive Voice Response
http://es.wikipedia.org/wiki/Interactive_Voice_Response
(artículo de la wikipedia sobre la IVR)

(47) Licencias de software libre
http://www.softwarelibre.ec/site/index.php?option=com_content&view=article&id=42&Itemid=148
(listado de las diferentes licencias de software libre)

(48) Lisp
http://es.wikipedia.org/wiki/Lisp
(artículo de la wikipedia sobre la familia de lenguajes Lisp)

(49) Junta de Andalucía
http://www.juntadeandalucia.es/index.html
(web oficial de la Junta de Andalucía)

(50) Dpkg
http://es.wikipedia.org/wiki/Dpkg
(artículo de la wikipedia sobre el programa dpkg)

(51) Gdebi
http://es.wikipedia.org/wiki/Gdebi
(artículo de la wikipedia sobre la herramienta gdebi)

(52) PulseAudio
http://www.pulseaudio.org/
(web oficial de PulseAudio)

(53) Juegos de caracteres
http://es.wikipedia.org/wiki/Codificaci%C3%B3n_de_caracteres
(artículo de la wikipedia sobre los juegos de caracteres)

(54) Iconv
http://www.gnu.org/software/libiconv/documentation/libiconv/iconv.1.html
(página con las especificaciones de Iconv)

(55) API C++ de Festival
http://www.cstr.ed.ac.uk/projects/festival/manual/festival_28.html
(página con las funciones de la API C++ de Festival)

(56) Sintaxis de SABLE
http://www.bell-labs.com/project/tts/sable.html
(página con la sintaxis completa de SABLE)

(57) Ejemplo de SABLE
http://www.cs.cmu.edu/~awb/festival_demos/sable.html
(ejemplo completo de las prestaciones de SABLE creado por Allan W. Black)

(58) Really Simple Syndication
http://es.wikipedia.org/wiki/RSS
(artículo de la wikipedia sobre la sindicación RSS)

(59) XSL
http://www.w3.org/TR/xslt
(especificaciones y documentación completa de XSL)

(60) Xalan
http://xml.apache.org/xalan-c/
(web con las especificaciones de Xalan para C++)

(61) Wget
http://es.wikipedia.org/wiki/GNU_Wget
(artículo de la wikipedia sobre Wget)

(63) Borland C++ Compiler
http://en.wikipedia.org/wiki/Borland_C%2B%2B
(artículo de la wikipedia sobre el antiguo compilador Borland C++, ahora CodeGear)

(64) Xalan C++ API Basic Usage Patterns
http://xml.apache.org/xalan-c/usagepatterns.html
(instrucciones de uso de la API C++ de Xalan en la web de Apache)

(65) Tutorial Xalan
http://rudeserver.com/xalan/xalan-manual.html
(tutorial básico de utilización de la API C++ de Xalan)

9_ PALABRAS CLAVE

TTS
Text-To-Speech
Texto-A-Habla
Texto-A-Voz
STT
Speech-To-Text
Habla-A-Texto
Voz-A-Texto
ASR
Automatic Speech Recognition
Reconocimiento de voz
Speech Synthesis
Síntesis del habla
Voz sintética
Festival
Flite
Festvox
RSS
RDF
XML
C++
Xalan
XSL
Markup language
Lenguaje de marcado
JSML
SSML
STML
SABLE
Accesibilidad
Deficiencia visual
Invidentes
DAISY
Asterisk
IVR